

Ю. В. КАПІТОНОВА
С. Л. КРИВИЙ
О. А. ЛЕТИЧЕВСЬКИЙ
Г. М. ЛУЦЬКИЙ
М. К. ПЕЧУРІН

ОСНОВИ ДИСКРЕТНОЇ МАТЕМАТИКИ



ПІДРУЧНИК

512/075
C 75

НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ КІБЕРНЕТИКИ ІМЕНІ В.М. ГЛУШКОВА
МІЖНАРОДНИЙ НАУКОВО-НАВЧАЛЬНИЙ ЦЕНТР
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА СИСТЕМ
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КІЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

**Ю.В. Капітонова, С.Л. Кривий,
О.А. Летичевський,
Г.М. Луцький, М.К. Печурін**

ОСНОВИ ДИСКРЕТНОЇ МАТЕМАТИКИ

ПІДРУЧНИК

КІЇВ НАУКОВА ДУМКА 2002

УДК 51.681.3517(075.8)

ББК 22.176я73

075

Мета підручника — систематичне викладення методів та засобів дискретної математики як інструментарію при обробці інформації в комп'ютерах. Висвітлюються основні математичні властивості тієї чи іншої теорії разом з фактами, необхідними для розв'язання задач. Матеріал подається на основі аксіоматичного методу і може служити основовою для таких спецкурсів, як бази даних і бази знань, теорія автоматів, системи штучного інтелекту, комп'ютерна алгебра, геометрія тощо.

Для студентів та аспірантів технічних вузів відповідних спеціальностей.

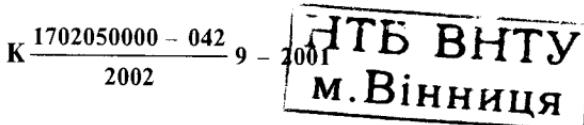
Затверджено до друку вченого радою
Інституту кібернетики НАН України

Допущено Міністерством освіти і науки України як підручник
(лист № 587 від 24. 04. 98)

415308

Редакція фізико-математичної
та технічної літератури

Редактор Т.С. Мельник



ISBN 966-00-0622-5

© Ю.В. Капітонова, С. Л. Кривий,
О.А. Летичевський, Г.М. Луцький,
М.К. Печурін, 2002

ПЕРЕДМОВА

Підручник присвячено видатному діячеві вітчизняної науки Віктору Михайловичу Глушкиву, Герою Соціалістичної праці, віце-президенту Академії наук України, професору, доктору фізико-математичних наук, дійсному члену Академії наук України і Академії наук СРСР, лауреату Ленінської премії, двічі лауреату Державної премії СРСР, двічі лауреату Державної премії України, кавалеру багатьох орденів та медалей, заслуженному діячеві науки і техніки, в доробку якого понад 800 наукових праць, у тому числі і 30 монографій. В.М.Глушкив є автором численних плідних наукових ідей, які не втратили свого значення і сьогодні.

Перша ідея — це ідея подання обчислювальної системи у вигляді системи алгебрологічних виразів, над якими можна виконувати ряд перетворень, а також у вигляді мережі алгоритмічних модулів. Ці модулі, в свою чергу, можна розглядати як мережі алгоритмічних модулів і т.п. Ідея алгебрологічних перетворень має універсальний характер і може втілюватися як в апаратурі, так і в програмі.

Друга ідея, що плідно використовується нині, — це ідея алгоритму очевидності. Суть цієї ідеї полягає в створенні спеціалізованої програми для автоматизованої обробки математичних текстів. Математичний текст подається як входні дані, а програма сама визначає, що саме розшифровується, а що ні, і подає запит, як визначити те, що вона не може розшифрувати.

Третя ідея — це ідея рекурсивної обчислювальної машини нової формашії з необмеженим нарощуванням пам'яті та швидкодії. Ця ідея частково була реалізована в макроконвеєрному обчислювальному комплексі. Машини з масовим паралелізмом долають граници класичного стилю організації обчислень і фактично становлять новий тип обчислювальних систем (такі системи зараз виготовляються в Японії та США). Макроконвеєрна організація обчислень була використана в суперЕОМ ЕС 1766, що дало можливість розв'язувати задачі з граничною швидкістю — близько 532 млн операцій за секунду. Макроконвеєрний принцип підтверджив, можна майже лінійно нарощувати швидкодію шляхом використання арифметичних процесорів.

Четверта ідея — це ідея створення штучного інтелекту як окремої системи, що визначається своїм розумом, який розміщується в базах знань, і системою перетворень, які динамічно самоорганізуються. Ця ідея має непересічний характер, оскільки дає можливість впроваджувати універсальні засоби роботи зі “знаннями”.

Ці ідеї та творчий доробок В.М. Глушкива були визнані міжнародною спільнотою, яка оцінила В.М. Глушкива як видатного організатора, що створив Інститут кібернетики в 1962 р. Інститут об'єднав багато людей, здатних займатися науковими дослідженнями. Зі школою В.М. Глушкива успішно співпрацювали школи інших видатних діячів вітчизняної науки, таких, як академіки В.С. Михалевич, О.І. Кухтенко, І.В. Сергієнко, В.М. Кунцевич, Б.Б. Тимофеєв, В.І. Скурихін, М.З. Згуровський, І.І. Ляшко, Б.М. Пшеничний, Н.З. Шор та багато інших.

Переходячи безпосередньо до характеристики даного підручника, необхідно зазначити ще одну ідею, яку розвивав В.М. Глущков. Основу сучасних швидкісних та якісних технологій обробки інформації становлять комп’ютери — від персональних до суперЕОМ. Подання інформації до ЕОМ дискретне, і її обробка складається з послідовностей елементарних перетворень тих чи інших інформаційних одиниць (слів, літер, цифр і т.п.). Отже, фундаментальною ідеєю щодо відображення реального світу в комп’ютері є ідея дискретизації об’єктів. Для ефективної роботи на комп’ютері необхідно навчитися будувати моделі реальних об’єктів та процесів їх перетворення. Досить часто такими моделями можуть бути конструкції дискретної математики, такі, як алгебра, формула, автомат, граф, алгоритм та ін. Останнім часом завдяки потребам комп’ютерної сфери дискретна математика розвивається досить динамічно. Її розділи поповнюються новими результатами, що виникають від своєрідної взаємодії різних галузей знань, які обслуговують комп’ютери. Літератури українською мовою з цієї тематики обмаль, а деякі розділи (наприклад, математична логіка) зовсім не висвітлені.

Автори ставлять за мету систематичне викладення засобів дискретної математики як інструментарію для подання та обробки інформації в комп’ютерах, а також алгебрологічних методів розв’язання задач. Зокрема, до першої частини ввійшли такі розділи: теорія множин і відношень, основні поняття загальної алгебри, елементи математичної логіки та теорія графів. При цьому висвітлюються основні математичні властивості тієї чи іншої теорії разом з фактами, які будуть потрібні й надалі.

На жаль, досить обмежений обсяг підручника в цілому впливну на повноту викладення матеріалу. Довелося скоротити число задач і вилучити доведення деяких теорем, а окрім розділи висвітлені досить стисло, хоч автори намагалися, щоб наведені факти давали повне розуміння суті тієї чи іншої теорії та її проблематики. Матеріал подається на основі аксіоматичного методу, і тому від читача вимагається певна математична культура та мінімальні знання з вищої алгебри та математичного аналізу.

Матеріал підручника може служити основою для таких спецкурсів, як бази даних і бази знань, теорія алгоритмів, комп’ютерна алгебра і комп’ютерна геометрія, текстові та графічні редактори тощо.

ЧАСТИНА I

МАТЕМАТИЧНІ ОСНОВИ

P o z d i l 1

МНОЖИНИ, ВІДНОШЕННЯ, КОМБІНАТОРИКА

Поняття множини — одне з основних, якщо не основне, поняття математики. Воно не має точного означення, і його слід віднести до аксіоматичних понять. Такими аксіоматичними поняттями, наприклад, в елементарній геометрії є поняття *точка, пряма, площа*.

Як правило, термін *множина* пояснюється за допомогою прикладів, а потім вказуються правила його використання в математичних застосуваннях. Останнє можна зробити на різних рівнях строгості. Детальне і строгое викладення теорії множин вимагало б скрупульозного аналізу логіки математичних суджень, а це — спеціальна самостійна тема, яка належить до області *основ математики*. Для наших цілей достатньо вибрати рівень так званої *інтуїтивної теорії множин*.

§1.1. МНОЖИНИ, ВІДНОШЕННЯ, ФУНКЦІЇ, ОПЕРАЦІЇ

1. ІНТУЇТИВНЕ ПОНЯТТЯ МНОЖИНИ. ОСНОВНІ ПРИНЦИПИ

Як було сказано вище, поняття множини належить до аксіоматичних понять математики, і точне його означення дати неможливо. Часто приймається формулювання інтуїтивного поняття множини Г. Кантора — основоположника цієї теорії:

“Довільне зібрання певних предметів нашої інтуїції чи інтелекту, які можна відрізняти один від одного і які уявляються як єдине ціле, називається *множиною*. Предмети, які входять до складу множини, називаються її *елементами*.”

Суттєвим пунктом канторівського розуміння множини є те, що зібрання предметів розглядається як один предмет (“уяв-

ляється як єдине ціле"). Основна увага тут переноситься з окремих предметів на зібрання предметів, які, в свою чергу, можна розглядати як предмети.

Що стосується "предметів нашої інтуїції чи інтелекту", то це формулювання дає значну свободу насамперед тим, що ніяк не обмежує природу предметів, з яких складається множина. Множина може складатися, наприклад, з людей, точок площини, простих чисел, планет Всесвіту. Зауважимо також, що кантопрівське формулювання множини дає змогу розглядати множини, елементи яких з певних причин точно визначити неможливо. У зв'язку з цим згадаємо, що елементи будь-якої нескінченної множини неможливо зібрати, навіть теоретично, в скінченну сукупність (з цією проблемою пов'язана філософська абстракція так званої актуальної нескінченності). Відомі також і скінченні множини, які мають таку ж міру невизначеності, як і всяка нескінчenna множина.

З'ясуємо, нарешті, зміст висловів: "які можна відрізнати один від одного" і "певні предмети". У першому випадку для будь-яких двох предметів, що розглядаються як елементи даної множини, повинна існувати можливість з'ясувати, чи різні ці предмети, чи однакові. У другому випадку, якщо задані деяка множина і який-небудь предмет, то можна визначити, чи цей предмет є елементом даної множини чи ні. Звідси випливає, що всяка множина повністю визначається своїми елементами. Ця кантопрівська вимога формулюється у вигляді аксіоми.

Інтуїтивний принцип об'ємності, або аксіома екстенсіональності. *Дві множини рівні тоді і тільки тоді, коли вони складаються з одних і тих же елементів. Рівність двох множин A і B позначають $A = B$.*

Отже, дві множини рівні, якщо кожний елемент однієї з них є елементом другої, і навпаки.

Множина називається *скінченою*, якщо вона складається із скінченного числа елементів. Запис $a \in A$ ($a \notin A$) означає, що a є (не є) елементом множини A . Однозначно визначена множина, елементами якої є a_1, a_2, \dots, a_n , познається $\{a_1, a_2, \dots, a_n\}$.

Приклади 1.1.1

1. Множина $\{a\}$ — так звана *одноелементна множина*, тобто множина, єдиним елементом якої є елемент a .

2. Множини $A = \{2, 4, 6\}$, $B = \{2, 6, 4\}$, $C = \{2, 2, 6, 4, 4\}$, $C' = \{2, 2, 4, 4, 6, 6\}$, ... рівні, оскільки складаються з одних і тих же елементів. Перша і друга множини відрізняються одна від одної порядком своїх елементів, а третя, четверта і т.д. — від

перших двох тим, що в ній елементи 2, 4 і 6 присутні в двох екземплярах кожний.

3. Множини $\{\{a, b\}, \{b, c\}\}$ і $\{a, b, c\}$ нерівні, оскільки перша складається з елементів $\{a, b\}$ і $\{b, c\}$, а друга — з елементів a, b, c .

4. Множини $\{\{1, 2\}\}$ і $\{1, 2\}$ нерівні, оскільки перша множина одноелементна, а друга — двохелементна.

Останній приклад показує, що необхідно розрізняти предмет і множину, єдиним елементом якої є цей предмет. ¹

Розглянемо детальніше ситуацію, яка має місце в прикладі 2. З принципу об'ємності випливає, що всі множини типу множин C, C' і т.д., які були наведені в цьому прикладі, рівні між собою і дорівнюють множині A .

Множина, яка складається з елементів деякої множини A так, що ці елементи можуть входити до складу цієї множини в якій завгодно кількості екземплярів, будемо називати **мультимножиною** множини A і позначати її $M(A)$. З точки зору теорії множин, множина і її мультимножина — це один і той же об'єкт, і вони можуть між собою не розрізнятися. Але часто, особливо коли йдеться про зображення множини в пам'яті обчислювальної машини, виникає потреба відрізняти мультимножину від множини.

Задання множини за допомогою фігурних дужок з явним переліком її елементів доцільне лише тоді, коли множина має невелику кількість елементів. Якщо ж множина має хоча і скінченну, але велику кількість елементів, таке задання множини досить громіздке, а у випадку нескінченної множини його застосування взагалі неможливе. Виникає питання: як задавати множини, які мають велике або нескінченне число елементів?

Відповідь пов'язана з поняттям властивості. Поки що обмежимося інтуїтивним поняттям властивості.

Під **властивістю** предмета x будемо розуміти таке розповідне речення, в якому щось стверджується відносно предмета x і яке можна характеризувати як істинне або хибне щодо x .

Приклади 1.1.2

1. Властивостями є такі записи:

- 1) З ділить x ;
- 2) $x < x$;
- 3) $x^2 = 2$;
- 4) $x^2 + 1 > 0$.

2. Вирази:

¹ Знак ¹ означає кінець прикладу чи групи прикладів.

- 5) для всіх x, y $xy = yx$;
- 6) існує таке x , що $2x < 0$,

не є властивостями, тому що їх не можна характеризувати як вірні чи хибні для певного x .

Нехай $P(x)$ означає деяку властивість, тоді $P(a)$ буде означати ту саму властивість, але із заміною x на a . Задання множини в термінах властивостей досягається за допомогою такого принципу.

Інтуїтивний принцип абстракції, або аксіома згортання. Всяка властивість $P(x)$ визначає деяку множину A за допомогою такої умови: елементами множини A є ті і тільки ті предмети a , які мають властивість P .

Згідно з принципом абстракції всяка властивість $P(x)$ визначає одну множину, яку позначають $\{a \mid P(a)\}$ і читають так: "множина всіх тих предметів a , що $P(a)$ ". Зауважимо, що властивість P може являти собою спосіб побудови елементів множини $\{a \mid P(a)\}$.

Нехай A — деяка множина, а $P(x)$ має вигляд $x \neq x$. Тоді множина $\{a \in A \mid P(a)\} = \{a \in A \mid a \neq a\}$, очевидно, не має елементів. Із принципу об'ємності випливає, що може існувати лише одна множина, яка не має елементів. Ця множина називається *пустою множиною* і позначається \emptyset .

Приклади множин 1.1.3

1. Множина натуральних чисел $\{0, 1, 2, \dots, n, \dots\}$, яку позначають буквою N . Цю множину можна задати за допомогою принципу згортання.

Нехай $A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ — множина з десяти елементів, яку будемо називати *алфавітом*. Всякий набір символів із цього алфавіту, записаних один за одним, назовемо *словом* в алфавіті A . Наприклад, $p = 0020034985700$ — слово в алфавіті A , а $p = 00a26543c$ не є словом в алфавіті A , оскільки до його запису входять символи a, c , що не належать множині A . Отже, p буде словом в алфавіті A тоді і тільки тоді, коли кожний символ цього слова належить алфавіту A .

Тоді множина $N = \{p = xy\dots z \mid x, y, \dots, z \in A\}$ становить множину натуральних чисел. Зауважимо, що слово $p = 00\dots 0$ і $p' = 0$ — одне і те ж число 0, так само, як і числа 00057 і 57. Отже, запис одного і того ж елемента множини N може бути різним. Для однозначності запису чисел необхідно поставити вимогу, щоб перший символ в слові p , яке складається більше ніж з одного символу, був відмінний від нуля, тобто

$N = \{ p = xy\dots z \mid x, y, \dots, z \in A \text{ і } x \neq 0 \text{ або } 0 \}.$

2. **Множина натуральних додатних чисел** $N^+ = \{1, 2, \dots, n, \dots\}$ — це така множина слів в алфавіті A , серед яких немає слова 0.

3. **Множина цілих чисел** $Z = \{\dots, -n, \dots, -2, -1, 0, 1, 2, \dots, n, \dots\}$ — це множина слів в алфавіті $B = \{-, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, таких, що $Z = \{p = xy\dots z \mid x, y, \dots, z \in B \text{ і } x \neq 0 \text{ і } y, \dots, z \neq - \text{ або } 0, \text{ якщо } p = 0\}$. Слово, першим символом якого є символ $-$, називається від'ємним, а слово, першим символом якого є символ $+$, називається додатним.

4. **Множина раціональних чисел** RC складається із множини всіх цілих чисел і всіх нескорочуваних раціональних дробів виду m/n , де $m, n \in Z$, $n \neq 0$, тобто $RC = \{m/n \mid m, n \in Z \text{ і } n \neq 0\}$. \blacktriangleleft

Підводячи підсумки, насамперед зазначимо, що множини задаються за допомогою або явного переліку своїх елементів (випадок скінченної множини з невеликою кількістю елементів), або властивостей, які задовольняють її елементи (властивість також може виражати спосіб побудови елементів множини). Задання множини будемо називати **ненадлишковим**, якщо кожний її елемент входить в дану множину в єдиному екземплярі, і **надлишковим**, якщо хоча б один елемент цієї множини входить до її складу більш як в одному екземплярі (випадок мультиможини).

Детальніше викладення основ теорії множин та її аксіоматику можна знайти в монографіях [32, 39, 72].

2. ОПЕРАЦІЇ НАД МНОЖИНАМИ. ЗАКОНИ ДЛЯ ОПЕРАЦІЙ

Введемо символи \Leftrightarrow , $\exists x$, $\forall x$, \Rightarrow , які надалі будуть служити для скорочення виразів “*тоді і тільки тоді, коли*”, “*існує x такий, що*”, “*для всякого x*” і “*слідує*” або “*випливає*” відповідно.

Множина A називається **підмножиною** множини B ($A \subseteq B$), якщо всі її елементи є також елементами множини B ($(A \subseteq B) \Leftrightarrow (a \in A \Rightarrow a \in B)$). При цьому множина B називається **надмножиною** множини A .

Тепер принцип об'ємності можна записати так:

$$A = B \Leftrightarrow (A \subseteq B \text{ і } B \subseteq A).$$

Вираз $A \subset B$ означає, що $A \subseteq B$ і A не дорівнює B ($A \neq B$). Якщо $A \subset B$, то множина A називається **власною підмножиною** множини B , а множина B — **власною надмножиною** множини A .

Покажемо, що пуста множина є підмножиною будь-якої множини A . Припустимо, що твердження $\emptyset \subseteq A$ хибне, тобто

існує хоча б один елемент x , що належить множині \emptyset , який не є елементом множини A . Але множина \emptyset не має елементів. Отже, твердження $\emptyset \subseteq A$ є істиною.

Приклади 1.1.4

1. Множина $A = \{a, b, c\}$ є власною підмножиною множини $B = \{a, b, c, d, e\}$.
2. Множина студентів юридичного факультету — підмножина множини всіх студентів університету.
3. Множина парних натуральних чисел є власною підмножиною множини всіх натуральних чисел.
4. Множина натуральних чисел є підмножиною множини всіх цілих чисел, а множина цілих чисел — підмножиною множини всіх раціональних чисел.

Нехай U — деяка множина. Тоді $B(U)$ — множина всіх підмножин множини U . У цьому випадку множину U називають **універсальною**, а множину $B(U)$ — **множиною-степенем** або **булеаном** множини U . Наприклад, якщо $U = \{1, 3, 5\}$, то $B(U) = \{\emptyset, \{1\}, \{3\}, \{5\}, \{1, 3\}, \{1, 5\}, \{3, 5\}, \{1, 3, 5\}\}$.⁴

Об'єднанням множин A і B називається множина, яка складається з тих і тільки тих елементів, які входять до складу хоча б однієї з цих множин. Одержанна множина позначається $A \cup B$, тобто $A \cup B = \{a \mid a \in A \text{ або } a \in B\}$.

Приклади 1.1.5

1. Нехай $A = \{1, 2, 3\}$, $B = \{1, 3, 4, 6\}$; тоді $A \cup B = \{1, 2, 3, 4, 6\}$.
2. Нехай Π — множина всіх парних натуральних чисел, а \mathbf{N} — множина всіх непарних натуральних чисел; тоді $\Pi \cup \mathbf{N} = \mathbf{N}$, де \mathbf{N} — множина всіх натуральних чисел.⁴

Перетином множин A і B називається множина, яка складається з елементів, що входять до складу як множини A , так і множини B . Одержанна множина позначається $A \cap B$, тобто $A \cap B = \{a \mid a \in A \text{ і } a \in B\}$. Якщо $A \cap B = \emptyset$, то множини A і B називаються такими, що не перетинаються.

Приклади 1.1.6

1. Нехай A, B, Π, \mathbf{N} — множини з попереднього прикладу; тоді:
 - a) $A \cap B = \{1, 3\}$;

- б) $P \cap H = \emptyset$;
- в) $N \cap H = H$;
- г) $N \cap P = P$.

2. Нехай A — множина прямих, які проходять через точку a деякої площини, а B — множина прямих, які проходять через точку c цієї ж площини. Тоді $A \cap B = \{l\}$, де l — пряма, яка проходить через точки a і c . \blacktriangleleft

Якщо множина A являє собою об'єднання підмножин $A_1, A_2, \dots, A_n, \dots$, то сукупність підмножин $\{A_1, A_2, \dots, A_n, \dots\}$ називається **покриттям** множини A . Якщо ж сукупність підмножин покриття множини A такі, що $A_i \cap A_j = \emptyset$ при $i \neq j$, то сукупність $\{A_1, \dots, A_n, \dots\}$ називається **роздиттям** множини A , а підмножини A_i — **класами** цього роздиття, $i = 1, 2, \dots, n, \dots$

Приклади покриття і роздиття множини 1.1.7

1. Нехай A — множина всіх студентів деякого вузу X , які його закінчили, а A_i — підмножина тих студентів вузу X , які закінчили i -й факультет цього вузу.

Оскільки не виключена можливість, що якась людина з множини студентів A закінчила кілька факультетів даного вузу, і така людина попадає в кілька відповідних підмножин сукупностей, то ясно, що сукупність підмножин A_1, A_2, \dots, A_k є покриттям множини A .

2. Якщо ж взяти сукупність всіх студентів вузу X , які навчаються в даний час, то сукупність студентів A_1, A_2, \dots, A_k є, очевидно, роздиттям множини всіх студентів даного вузу, які навчаються в даний час. \blacktriangleleft

Різницею множин A і B називається множина $B \setminus A = \{a \mid a \in B \text{ і } a \notin A\}$. Очевидно, що $B \setminus A = B \setminus (A \cap B)$. Якщо $A \subseteq B$, то $B \setminus A$ називається **доповненням** множини A в множині B і позначається A'_B або просто A' , коли B можна визначити із контексту.

Симетричною різницею множин A і B називається множина $A \Delta B = (A \setminus B) \cup (B \setminus A)$.

Приклади 1.1.8

1. Нехай $A = \{1, 2, 3\}$, $B = \{1, 3, 4, 5\}$. Тоді $B \setminus A = \{1, 3, 4, 5\} \setminus \{1, 2, 3\} = \{4, 5\} = B \setminus (A \cap B) = \{1, 3, 4, 5\} \setminus \{1, 3\} = \{4, 5\}$.

2. Множина $N \setminus P = H$, тобто $N \setminus P$ являє собою множину всіх непарних натуральних чисел. Навпаки, $N \setminus H = P$. \blacktriangleleft

Введені операції називають **теоретико-множинними операціями**. Їх можна ілюструвати графічно за допомогою так званих **діаграм Венна** (рис. 1.1.1.). На цих діаграмах множини-аргументи зображуються у вигляді областей площини, а результат виконання операції — у вигляді заштрихованої області.

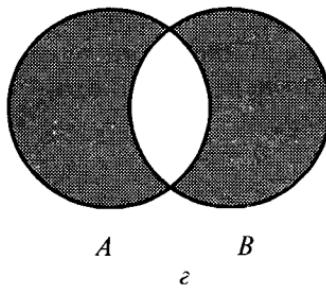
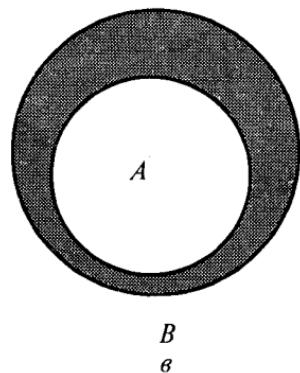
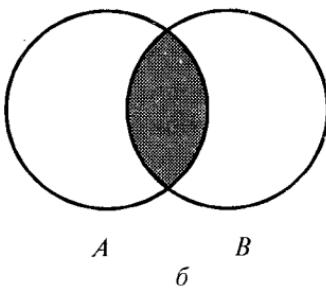
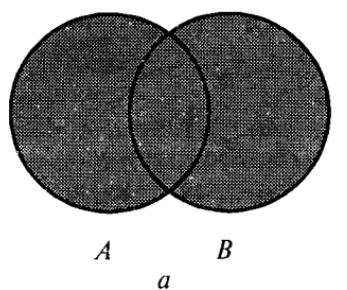


Рис 1.1.1. Діаграми Венна:

a — діаграма для $A \cup B$; *b* — діаграма для $A \cap B$;
c — діаграма для A' ; *d* — діаграма для $A + B$

З наведеної діаграми для операції $A + B$ очевидним чином випливає така рівність: $A + B = (A \cup B) \setminus (A \cap B)$.

Введені операції об'єднання, перетину і доповнення задовільняють певні закони. Має місце така теорема.

Теорема 1.1.1. Для довільних підмножин A, B, C деякої універсальної множини U мають місце такі тотожності:

- M1) $A \cup B = B \cup A, A \cap B = B \cap A$ (комутативність);
- M2) $A \cup (B \cup C) = (A \cup B) \cup C,$
 $A \cap (B \cap C) = (A \cap B) \cap C$ (асоціативність);
- M3) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C),$
 $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ (дистрибутивність);
- M4) $A \cup A' = U, A \cap A' = \emptyset;$
- M5) $A \cup \emptyset = A, A \cap U = A.$

Д о в е д е н н я. Покажемо справедливість другого співвідношення з М2: $A \cap (B \cap C) = (A \cap B) \cap C$.

Нехай $a \in A \cap (B \cap C) \Rightarrow a \in A, a \in B, a \in C \Rightarrow a \in (A \cap B)$ і $a \in C \Rightarrow a \in (A \cap B) \cap C$ і $A \cap (B \cap C) \subseteq (A \cap B) \cap C$.

Одержання оберненого включення виконується аналогічно.

Покажемо справедливість першого співвідношення із М3. З одного боку, оскільки $(B \cap C) \subseteq B$, то $A \cup (B \cap C) \subseteq A \cup B$. Аналогічно $(B \cap C) \subseteq C$ і $A \cup (B \cap C) \subseteq A \cup C$. Значить, $A \cup (B \cap C) \subseteq (A \cup B) \cap (A \cup C)$.

З іншого боку, якщо $a \in (A \cup B) \cap (A \cup C)$, то $a \in A \cup B$ і $a \in A \cup C$. Якщо $a \in A$, то $a \in A \cup (B \cap C)$. А якщо $a \notin A$, то $a \in B$ і $a \in C$ і тоді $a \in B \cap C$. Отже, $(A \cup B) \cap (A \cup C) \subseteq A \cup (B \cap C)$. Разом з отриманим раніше включенням маємо потрібну рівність.

Рівності доведені.

Решта співвідношень доводиться аналогічно, і їх доведення пропонуються читачеві як вправи (див. вправу 9 в кінці параграфа).

Теорема доведена.

Використовуючи закони асоціативності для операцій об'єднання і перетину множин, можна ввести такі скорочення.

Вирази

$$A_1 \cup A_2 \cup \dots \cup A_n = \bigcup_{i=1}^n A_i$$

означають об'єднання множин A_1, A_2, \dots, A_n , тобто сукупність тих елементів, які є елементами хоча б однієї з множин A_1, A_2, \dots, A_n , а вирази

$$A_1 \cap A_2 \cap \dots \cap A_n = \bigcap_{i=1}^n A_i$$

означають перетин множин, тобто сукупність тих предметів, які є елементами кожної з множин A_1, A_2, \dots, A_n .

Окрім тотожностей, які були наведені в теоремі 1.1.1, існують і інші корисні тотожності для операцій над множинами.

Теорема 1.1.2. Для будь-яких підмножин A і B деякої універсальної множини U справедливі такі тотожності:

$$\text{M6}) A \cup A = A, \quad A \cap A = A;$$

$$\text{M7}) A \cup (A \cap B) = A, \quad A \cap (A \cup B) = A;$$

$$\text{M8}) A \cup U = U, \quad A \cap \emptyset = \emptyset;$$

$$\text{M9}) (\emptyset)' = U, \quad U' = \emptyset, \quad (A')' = A;$$

$$\text{M10}) (A \cup B)' = A' \cap B', \quad (A \cap B)' = A' \cup B'.$$

Д о в е д е н н я. Покажемо, наприклад, справедливість тотожностей M6 і M8.

В силу законів М1—М5 можемо записати:

$$\begin{aligned} A &= A \cup \emptyset = A \cup (A \cap A') = (A \cup A) \cap (A \cup A') = \\ &= (A \cup A) \cap U = A \cup A, \\ A &= A \cap U = A \cap (A \cup A') = (A \cap A) \cup (A \cap A') = \\ &= (A \cap A) \cup \emptyset = A \cap A. \end{aligned}$$

Далі, використовуючи тільки що встановлені тотожності, одержуємо

$$\begin{aligned} A \cup U &= A \cup (A \cup A') = (A \cup A) \cup A' = A \cup A' = U, \\ A \cap \emptyset &= A \cap (A \cap A') = (A \cap A) \cap A' = A \cap A' = \emptyset. \end{aligned}$$

Доведення решти тотожностей пропонуються читачеві як вправи (див. також розділ 2 “Булеві алгебри”).

Теорема доведена.

Користуючись тотожностями, наведеними в теоремах 1.1.1 і 1.1.2, можна спрощувати складні вирази, які містять у собі множини, аналогічно тому, як проводяться спрощення виразів в елементарній алгебрі. Розглянемо приклади.

Приклади 1.1.9

$$\begin{aligned} 1. (A \cap B \cap C) \cup (A' \cap B \cap C) \cup B' \cup C' &= \\ &= [(A \cup A') \cap B \cap C] \cup B' \cup C' = (U \cap B \cap C) \cup B' \cup C' = \\ &= (B \cap C) \cup B' \cup C' = (B \cap C) \cup (B \cap C)' = U. \\ 2. (A \cap B \cap C \cap D') \cup (A' \cap C) \cup (B' \cap C) \cup (C \cap D) &= \\ &= (A \cap B \cap C \cap D') \cup [(A' \cup B' \cup D) \cap C] = \\ &= [(A \cap B \cap D') \cup (A \cap B \cap D')'] \cap C = U \cap C = C. \\ 3. (A \cap B')' \cup B &= (A' \cup B'') \cup B = A' \cup (B \cup B) = A' \cup B. \end{aligned}$$

3. ДЕКАРТОВИЙ ДОБУТОК МНОЖИН. ВІДНОШЕННЯ

Нехай A і B — дві множини. Розглянемо множину $C = \{(a, b) | a \in A, b \in B\}$. Ця множина називається **декартовим добутком множин A і B** та позначається $A \times B$. Якщо множини A і B скінченні і складаються відповідно з m і n елементів, то очевидно, що C складається з mn елементів.

Самостійний інтерес викликає випадок, коли множина A дорівнює B : $A = B$. Для розгляду цього випадку введемо поняття упорядкованої пари.

Упорядкованою парою елементів множини A назовемо об'єкт (a, a') , що складається з двох, не обов'язково різних, елементів — a і a' множини A , для яких вказано, котрий потрібно вважати першим, а котрий — другим. Так, якщо $A = \{1, 2, 3, 4, 5\}$, то упорядковані пари $(3, 4)$ і $(4, 3)$ слід вважати різними, оскільки в

першій парі першим елементом є 3, другим — 4, а в другій парі — навпаки. Упорядкованими парами є також пари (1, 1), (2, 2), (3, 3), (4, 4), (5, 5).

Множина $C = \{(a, a') \mid a, a' \in A\}$ всіх упорядкованих пар (a, a') елементів із множини A називається **декартовим квадратом множини A** і позначається A^2 .

Поняття упорядкованої пари можна розширити на упорядковані трійки елементів (a_1, a_2, a_3) , упорядковані четвірки (a_1, a_2, a_3, a_4) і т.д. В загалі, упорядкована n -ка елементів із множини A — це n не обов'язково різних елементів із A , заданих у певній послідовності. Якщо

$$(a_1, a_2, \dots, a_n) = (a'_1, a'_2, \dots, a'_n),$$

то $a_1 = a'_1, a_2 = a'_2, \dots, a_n = a'_n$.

Для того щоб відрізняти упорядковані пари, трійки і т.д. від неупорядкованих, введемо таке позначення: якщо A — деяка множина, то $A^{(2)}, A^{(3)}, \dots$ будуть відповідно означати множину неупорядкованих пар елементів, трійок елементів і т.д. із множини A , тобто $A^{(2)}$ являє собою множину всіх двохелементних підмножин множини A , $A^{(3)}$ — множину всіх трьохелементних підмножин множини A і т.д.

Наведені вище означення декартового добутку двох множин і декартового квадрата множини можна звичайним способом узагальнити і на випадок довільної скінченної сукупності множин.

Декартовим добутком $A_1 \times A_2 \times \dots \times A_n$ множин A_1, A_2, \dots, A_n називається сукупність послідовностей (тобто сукупність упорядкованих n -ок елементів) вигляду (a_1, a_2, \dots, a_n) , де $a_i \in A_i, 1 \leq i \leq n$.

Елементи декартового добутку називають іще **кортежами**. Довільна підмножина множини $A_1 \times A_2 \times \dots \times A_n$ називається **відношенням**, заданим або визначенням на множинах A_1, A_2, \dots, A_n . Якщо $A_1 = A_2 = \dots = A_n = A$, то декартовий добуток $A_1 \times A_2 \times \dots \times A_n$ називається **декартовим добутком n -го степеня множини A** (A^n), а відношення R , задане на множинах A_1, A_2, \dots, A_n , — **n -арним відношенням** на множині A .

Коли $(a_1, a_2, \dots, a_n) \in R$, то говорять, що елементи a_i ($i = 1, 2, \dots, n$) знаходяться між собою у відношенні R або відношення R істинне для a_1, a_2, \dots, a_n . Якщо $(a_1, a_2, \dots, a_n) \notin R$, то вважають, що R хибне для a_1, a_2, \dots, a_n . При $n = 1$ відношення називається **унарним**, при $n = 2$ — **бінарним**, при $n = 3$ — **тернарним** і т.д.

Оскільки відношення, задані на A_1, A_2, \dots, A_n , — підмножини множини $A_1 \times A_2 \times \dots \times A_n$, то для них визначені операції об'єднання, перетину, різниці і доповнення:

$$(a_1, a_2, \dots, a_n) \in R \cup R_1 \Leftrightarrow (a_1, a_2, \dots, a_n) \in R \text{ або}$$

- $$(a_1, a_2, \dots, a_n) \in R_1;$$
- $$(a_1, a_2, \dots, a_n) \in R \cap R_1 \Leftrightarrow (a_1, a_2, \dots, a_n) \in R \text{ i } (a_1, a_2, \dots, a_n) \in R_1;$$
- $$(a_1, a_2, \dots, a_n) \in R \setminus R_1 \Leftrightarrow (a_1, a_2, \dots, a_n) \in R \text{ i } (a_1, a_2, \dots, a_n) \notin R_1;$$
- $$(a_1, a_2, \dots, a_n) \in R' \text{ в } A_1 \times A_2 \times \dots \times A_n \Leftrightarrow (a_1, a_2, \dots, a_n) \notin R.$$

Часто R' позначають $\neg R$ і називають **запереченням** відношення R .

Досить часто як в теорії, так і на практиці використовуються бінарні відношення. Якщо R — бінарне відношення, то запис aRb означає, що $(a, b) \in R$, тобто що R істинне для a, b .

Нехай $R \subseteq A \times B$, $R_1 \subseteq B \times C$. Відношення R^{-1} , задане на множині $B \times A$, називають **оберненим** до R , якщо $R^{-1} = \{(b, a) \mid aRb\}$, а відношення $R * R_1$, задане на множині $A \times C$, — **добутком** або **суперпозицією** відношень R і R_1 , якщо $R * R_1 = \{(a, c) \mid a \in A, c \in C \text{ i } (\exists b \in B) (aRb \text{ i } bR_1c)\}$.

Слід зазначити, що операція множення відношень може бути і невизначеною, якщо в множині B для заданих елементів a із A і c із C не існує відповідного елемента b . Але, якщо $A = B = C$, то ця операція завжди визначена.

Розглянемо деякі корисні властивості бінарних відношень.

Теорема 1.1.3. Якщо R, R_1, R_2 — бінарні відношення, задані на множині A , то:

- 1) $(R_1 \cup R_2) * R = R_1 * R \cup R_2 * R; R_1 \subseteq R_2 \Rightarrow R_1 * R \subseteq R_2 * R;$
- 2) $(R^{-1})^{-1} = R; R \subseteq R_1 \Rightarrow R^{-1} \subseteq R_1^{-1};$
- 3) $(R * R_1)^{-1} = (R_1^{-1}) * (R^{-1});$
- 4) $(R \cap R_1)^{-1} = (R^{-1}) \cap (R_1^{-1});$
- 5) $(R * R_1) * R_2 = R * (R_1 * R_2).$

Д о в е д е н н я. 1. Якщо $(a, b) \in (R_1 \cup R_2) * R$, то існує елемент $c \in A$, такий, що $(a, c) \in R_1 \cup R_2$ і $(c, b) \in R$. Значить, $(a, c) \in R_1$ або $(a, c) \in R_2$ і $(c, b) \in R$. Звідси маємо, що $(a, b) \in R_1 * R$ або $(a, b) \in R_2 * R$, тобто $(a, b) \in R_1 * R \cup R_2 * R$. Обернене включення доводиться аналогічно.

Друга частина твердження випливає з того, що коли $R_1 \subseteq R_2$, то $R_1 \cup R_2 = R_2$, звідки маємо (в силу доведеного вище), що $(R_1 \cup R_2) * R = R_1 * R \cup R_2 * R = R_2 * R$, тобто $R_1 * R \subseteq R_2 * R$.

2. $(a, b) \in R^{-1} \Leftrightarrow (b, a) \in (R^{-1})^{-1} \Leftrightarrow (b, a) \in R$, звідки випливає, що $R = ((R^{-1})^{-1})$.

Для доведення другої частини зауважимо, що $(a, b) \in R \Leftrightarrow (b, a) \in R^{-1} \Rightarrow (a, b) \in R \Rightarrow (a, b) \in R_1 \Rightarrow (b, a) \in R^{-1} \Rightarrow (b, a) \in R_1^{-1}$, тобто $R^{-1} \subseteq R_1^{-1}$.

3. $(a, b) \in (R * R_1)^{-1} \Leftrightarrow (b, a) \in (R * R_1) \Rightarrow (\exists c \in A) (b, c) \in R \text{ i } (c, a) \in R_1$. Але тоді $(c, b) \in (R^{-1})$ і $(a, c) \in (R_1^{-1}) \Rightarrow (a, b) \in$

$\in (R_l^{-1}) * (R^{-1})$, тобто $(R * R_l)^{-1} \subseteq (R_l^{-1}) * (R^{-1})$. Обернене включення доводиться аналогічно.

4. $(a, b) \in (R \cap R_l)^{-1} \Leftrightarrow (b, a) \in R \cap R_l \Leftrightarrow (b, a) \in R \text{ i } (b, a) \in R_l \Leftrightarrow (a, b) \in (R^{-1}) \text{ i } (a, b) \in (R_l^{-1})$, тобто $(R \cap R_l)^{-1} = (R^{-1}) \cap (R_l^{-1})$.

5. Нехай $(a, d) \in (R * R_l) * R_2$, тоді існує $c \in A$, такий, що $(a, c) \in R * R_l \text{ i } (c, d) \in R_2$. Отже, існує b такий, що $(a, b) \in R$, $(b, c) \in R_l$ і $(c, d) \in R_2$, а це означає, що $(b, d) \in R_l * R_2$ і $(a, d) \in R * (R_l * R_2)$, тобто $(R * R_l) * R_2 \subseteq R * (R_l * R_2)$. Обернене включення доводиться аналогічно.

Теорема доведена.

4. ПРИКЛАДИ ВІДНОШЕНЬ

Розглянемо приклади найбільш важливих відношень.

1. Відношення тотожності. Бінарне відношення *тотожності*, задане на множині A , складається із всіх пар, що мають вигляд (a, a) , де $a \in A$, і позначається i_A або просто i , якщо A фіксовано. Такі пари (a, a) називають *діагональними*, а відношення i_A — *діагональю*. Очевидно, що для будь-якого бінарного відношення R , визначеного на множині A , справедлива рівність $i_A * R = R * i_A = R$.

2. Рефлексивне відношення. Бінарне відношення R , задане на множині A , називається *рефлексивним*, якщо $i_A \subseteq R$, тобто коли воно включає діагональ.

Прикладом рефлексивних відношень можуть служити такі бінарні відношення:

- пряма x паралельна прямій y в площині z ;
- студент x — ровесник студента y .

Дійсно, в першому випадку з елементарної геометрії відомо, що дві прямі, які лежать в одній площині, паралельні, якщо вони або збігаються, або не мають жодної спільної точки, скільки б їх не продовжували. Оскільки пряма x збігається сама з собою, то пара (x, x) належить даному відношенню.

У другому випадку очевидно, що кожний студент — сам собі ровесник.

3. Іррефлексивні відношення. Бінарне відношення R називається *іррефлексивним*, якщо aRa не має сенсу. Наприклад, відношення строгого порядку $x < x$ на множині дійсних чи раціональних чисел не має сенсу, тому що воно завжди хибне.

4. Симетричні відношення. Бінарне відношення R , задане на множині A , називається **симетричним**, якщо $aRb \Rightarrow bRa$ ($R \subseteq R^{-1}$).

Прикладом симетричних відношень можуть служити такі бінарні відношення:

- пряма x перпендикулярна до прямої y в площині z ;
- студент x є сусідом по парті студента y .

Дійсно, у першому випадку з елементарної геометрії відомо: якщо пряма x перпендикулярна до прямої y , то і пряма y перпендикулярна до прямої x .

У другому випадку всякий студент може впевнитися, що коли студент y є його сусідом, то сусідом студента y є він сам. Зауважимо, що наведені відношення не є рефлексивними.

5. Транзитивні відношення. Бінарне відношення R , задане на множині A , називається **транзитивним**, якщо з aRb і bRc випливає aRc ($R^2 \subseteq R$).

Прикладом транзитивних відношень можуть служити такі бінарні відношення:

- місто x зв'язане з містом y шосейною дорогою;
- студент x є ровесником студента y ;
- трикутник x подібний трикутнику y ;
- дійсне число x більше дійсного числа y .

Отже, у першому випадку, якщо між містами x і y є шосейна дорога і між містами y , z також є шосейна дорога, то ясно, що між містами x , z теж є шосейна дорога, яка, наприклад, пролягає через місто y .

В останніх трьох випадках транзитивність очевидна.

6. Антисиметричні відношення. Бінарне відношення R , задане на множині A , називається **антисиметричним**, якщо з aRb і bRa випливає $a = b$ ($R \cap R^{-1} \subseteq i_A$).

Прикладом антисиметричного відношення може служити бінарне відношення включення для множин, тобто відношення “множина A є підмножиною множини B ”.

Справді, якщо $A \subseteq B$ і $B \subseteq A$, то з аксіоми об'ємності випливає, що множини A і B складаються з одних і тих же елементів, тобто $A = B$.

7. Відношення еквівалентності. Бінарне відношення R , задане на множині A , називається **відношенням еквівалентності** або просто **еквівалентністю** на A , якщо для будь-яких елементів a, b, c із A справедливі такі властивості:

- aRa $(i_A \subseteq R)$ (рефлексивність);
- $aRb \Rightarrow bRa$ $(R \subseteq R^{-1})$ (симетричність);
- aRb і $bRc \Rightarrow aRc$ $(R^2 \subseteq R)$ (транзитивність),

де i_A — відношення тотожності, а $R^2 = R * R$.

Неважко показати, що умови “а”, “б”, “в” еквівалентні таким: $i_A \subseteq R$, $R = (R^{-1})$, $R^2 = R$ (див. вправу 29 в кінці параграфа).

Відношення еквівалентності, задане на множині A , тісно пов’язане з розбиттям множини A на класи. Цей зв’язок виражається такими твердженнями.

Лема 1.1.1. *Всяке розбиття множини A на класи визначає на множині A відношення еквівалентності.*

Д о в е д е н н я. Нехай $a, b \in A$, покладемо $aRb \Leftrightarrow a \text{ і } b \text{ лежать в одному класі розбиття}$. Покажемо, що одержане бінарне відношення є відношенням еквівалентності. Для цього треба довести, що воно рефлексивне, симетричне та транзитивне. Дійсно, оскільки a лежить у деякому класі розбиття, то aRa , тобто воно рефлексивне.

Нехай K — деякий клас розбиття і $a, b \in K$. Тоді і $b, a \in K$, тобто $aRb \Rightarrow bRa$. Симетричність доведено.

Із aRb і bRc випливає $a, b, c \in K$. Звідси aRc , що потрібно було довести.

Лема 1.1.2. *Всяке відношення еквівалентності R , визначене на множині A , задає розбиття множини A на класи.*

Д о в е д е н н я. Назвемо класом елемента a множину $K(a) = \{x \in A \mid aRx\}$. Із рефлексивності відношення R випливає, що $a \in K(a)$, тобто система класів $K(a)$ ($a \in A$) покриває всю множину A . Далі, симетричність відношення R показує, що коли $b \in K(a)$, то $a \in K(b)$, а транзитивність відношення R приводить до того, що коли $b \in K(a)$, то із $c \in K(b)$ випливає $c \in K(a)$, тобто $K(b) \subseteq K(a)$. Але якщо $a \in K(b)$, то $K(a) \subseteq K(b)$, значить, $K(a) = K(b)$. Звідси випливає, що кожний клас визначається будь-яким своїм елементом. Якщо $K(a) \cap K(b) \neq \emptyset$, то існує такий елемент c із цього перетину, що cRa і cRb , і тоді $K(a) = K(b) = K(c)$, тобто класи $K(a)$ і $K(b)$ збігаються. Таким чином, система всіх різних класів виду $K(a)$ є розбиттям множини A , що і потрібно було довести.

Очевидно, що перехід від розбиття S множини A , яке визначається відношенням R , до відношення еквівалентності R , а після цього перехід від відношення R до розбиття множини A , знову приводить до розбиття S . Таким чином, справедлива така теорема.

Теорема 1.1.4. *Між розбиттями множини на класи і відношеннями еквівалентності, заданими на цій множині, існує взаємно однозначна відповідність [42].*

Якщо R — еквівалентність на A , то класи розбиття, визначені відношенням R , називають **класами еквівалентності** відношення R , а множину класів — **фактор-множиною** множини A і позначають A/R . Число класів еквівалентності відношення еквівалент-

ності R називається *індексом* множини A . Коли число класів еквівалентності скінченне, то множина A називається *множиною скінченного індексу*.

З'ясуємо, які з операцій теорії множин зберігають властивість відношення бути відношенням еквівалентності, а які ні.

Теорема 1.1.5. Якщо R, R_1 — відношення еквівалентності, задані на множині A , то:

1) R^{-1} — відношення еквівалентності на A ;

2) $R * R_1$ — відношення еквівалентності на A тоді і тільки тоді, коли $R * R_1 = R_1 * R$, тобто коли відношення R і R_1 можна міняти місцями;

3) $R \cap R_1$ — відношення еквівалентності на A ;

4) R' не є відношенням еквівалентності на A .

Доведення. 1. Оскільки R — відношення еквівалентності, то $R^{-1} = R$ і, значить, R^{-1} — теж відношення еквівалентності.

2. Якщо $R * R_1$ — відношення еквівалентності, то за доведеним вище $(R * R_1)^{-1} = (R_1 * R)^{-1}$ — теж відношення еквівалентності і за теоремою 1.1.3 $(R * R_1)^{-1} = (R_1^{-1}) * (R^{-1}) = R_1 * R$.

Навпаки, якщо $R * R_1 = R_1 * R$, то із того, що xRx і xR_1x , випливає $xR * R_1x$, і, отже, $R * R_1$ рефлексивне. Далі, із того, що $(R * R_1)^{-1} = (R_1^{-1}) * (R^{-1}) = R_1 * R = R * R_1$, випливає симетричність $R * R_1$. І, нарешті, $(R * R_1) * (R * R_1) = R * R * R_1 * R_1 = R * R_1$, оскільки їх можна міняти місцями і множення відношень асоціативне (див. теорему 1.1.3). Отже, $R * R_1$ транзитивне.

3. Оскільки R і R_1 — еквівалентності на A , то $i_A \subseteq R$ і $i_A \subseteq R_1$, звідки випливає, що $i_A \subseteq R \cap R_1$.

За теоремою 1.1.3 і через те, що R і R_1 — еквівалентності, маємо $(R \cap R_1)^{-1} = (R^{-1}) \cap (R_1^{-1}) = R \cap R_1$. Значить, відношення $R \cap R_1$ рефлексивне і симетричне. Покажемо його транзитивність.

Нехай $(a, b) \in R \cap R_1$ і $(b, c) \in R \cap R_1$. Тоді $(a, b) \in R$, $(a, b) \in R_1$ і $(b, c) \in R$, $(b, c) \in R_1$. Із $(a, b) \in R$, $(b, c) \in R$ випливає, що $(a, c) \in R$, а із $(a, b) \in R_1$, $(b, c) \in R_1$, — що $(a, c) \in R_1$. Звідси робимо висновок, що $(a, c) \in (R \cap R_1)$, тобто $R \cap R_1$ транзитивне.

4. Із того, що R — еквівалентність, випливає, що $i_A \subseteq R$, але тоді i_A не може бути підмножиною множини R' , тобто R' не рефлексивне і, отже, не є відношенням еквівалентності.

Теорема доведена.

Об'єднання відношень еквівалентності, загалом, не завжди є відношенням еквівалентності, як показує така теорема.

Теорема 1.1.6. *Об'єднання $R \cup R_1$ відношень еквівалентності R і R_1 є еквівалентністю тоді і тільки тоді, коли перетин будь-якого класу еквівалентності по R з будь-яким класом еквівалентності по R_1 або збігається з одним із них, або пустий. Якщо $R \cup R_1$ — еквівалентність, то $R \cup R_1 = R * R_1$ [50].*

8. Замикання відношень. Нехай R — деяке бінарне відношення на множині A :

a) **рефлексивним замиканням** R_1 відношення R називається відношення $R \cup i$, де i — відношення тотожності на A (діагональ);

b) **симетричним замиканням** R_s відношення R називається відношення $R \cup R^{-1}$, тобто якщо $(a, b) \in R$, то $(a, b) \in R_s$ і $(b, a) \in R_s$;

b) **транзитивним замиканням** R_t відношення R називається відношення $R_t = R \cup R^2 \cup R^3 \cup \dots \cup R^n \cup \dots$, тобто $(a, b) \in R_t$ тоді і тільки тоді, коли існують елементи $a_1 = a, a_2, \dots, a_n = b \in A$, такі, що $a_1 R a_2, a_2 R a_3, \dots, a_{n-1} R a_n$.

Якщо деяке відношення містить своє симетричне, рефлексивне і транзитивне замикання, то воно є відношенням еквівалентності, і навпаки.

П р и к л а д 1.1.10

Нехай $A = \{a_1, a_2, a_3, a_4\}$ і $R = \{(a_1, a_3), (a_3, a_4), (a_4, a_2), (a_3, a_3)\}$.

Тоді

$$R_i = \{(a_1, a_3), (a_3, a_4), (a_4, a_2), (a_3, a_3), (a_1, a_1), (a_2, a_2), (a_4, a_4)\},$$

$$R_s = \{(a_1, a_3), (a_3, a_1), (a_3, a_4), (a_4, a_3), (a_4, a_2), (a_2, a_4), (a_3, a_3)\},$$

$$R_t = \{(a_1, a_3), (a_3, a_4), (a_4, a_2), (a_3, a_3), (a_1, a_4), (a_1, a_2), (a_3, a_2)\}. \blacksquare$$

9. Відображення і операції. Відношення F , задане на множинах A_1, A_2, \dots, A_n, B , називається **функціональним**, якщо для будь-якого елемента (a_1, a_2, \dots, a_n) із $A_1 \times A_2 \times \dots \times A_n$ існує не більше одного елемента b із B , такого, що $(a_1, a_2, \dots, a_n, b) \in F$. Якщо такий елемент b із B існує для деякого (a_1, a_2, \dots, a_n) , то він позначається $F(a_1, a_2, \dots, a_n)$ і записується так: $b = F(a_1, a_2, \dots, a_n)$.

Нехай $F^{-1}(b) = \{(a_1, a_2, \dots, a_n) \in A_1 \times A_2 \times \dots \times A_n \mid F(a_1, a_2, \dots, a_n) = b\}$ і $\text{Dom}(F) = \bigcup_{b \in B} F^{-1}(b)$. Очевидно, що для всякого функціонального відношення F , заданого на A_1, A_2, \dots, A_n, B , виконується включення $\text{Dom}(F) \subseteq A_1 \times A_2 \times \dots \times A_n$.

У випадку, коли $\text{Dom}(F) = A_1 \times A_2 \times \dots \times A_n$, відношення F називається **повністю визначенім**, а коли $\text{Dom}(F) \subset A_1 \times A_2 \times \dots \times A_n$, — **частково визначенім** або просто **частковим**.

Відношення F , задане на множинах A_1, A_2, \dots, A_n, B , нази-

вається **відображенням** або **функцією** із $A_1 \times \dots \times A_n$ в B ($F: A_1 \times \dots \times A_n \rightarrow B$), якщо F функціональне і повністю визначене. Відношення F називається **частковим** відображенням або **частковою функцією**, якщо F функціональне і часткове. Число n називається **аристою функції** F .

Якщо $F: A_1 \times A_2 \times \dots \times A_n \rightarrow B$ і існує b із B , такий, що $F(a_1, a_2, \dots, a_n) = b$, то елемент b називають **образом** елемента (a_1, a_2, \dots, a_n) при відображені F , а елемент (a_1, a_2, \dots, a_n) — **прообразом** елемента b . Множину

$$F^{-1}(b) = \{(a_1, a_2, \dots, a_n) \mid F(a_1, a_2, \dots, a_n) = b\},$$

введену раніше, називають **повним прообразом** елемента b в множині $A_1 \times A_2 \times \dots \times A_n$.

Відношення $F: A_1 \times A_2 \times \dots \times A_n \rightarrow B$ називають **відображенням на** B тоді і тільки тоді, коли $\forall b \in B F^{-1}(b) \neq \emptyset$.

Відображення F множини $A_1 \times A_2 \times \dots \times A_n$ на множину B називається **взаємно однозначним відображенням** або **взаємно однозначною відповідністю** тоді і тільки тоді, коли обернене до відношення F відношення F^{-1} є відображенням B на $A_1 \times A_2 \times \dots \times A_n$.

Нам відома операція добутку відношень. Оскільки відображення — це відношення спеціального виду, то з'ясуємо, що собою являє добуток відображень.

Нехай $F_1: A \rightarrow B$, а $F: B \rightarrow C$ — деякі відображення. За визначенням добутку відношень маємо: $(a, c) \in F_1 * F \Leftrightarrow$ існує елемент $b \in B$, такий, що $(a, b) \in F_1$ і $(b, c) \in F$, тобто $F_1(a) = b$ і $F(b) = c$, або $F(F_1(a)) = c$ згідно з прийнятими вище позначеннями. Таким чином, добуток відображень являє собою добре відому операцію суперпозиції функцій.

Покажемо, що добуток відображень є асоціативною операцією.

Нехай $F: A \rightarrow B$, $F_1: B \rightarrow C$, $F_2: C \rightarrow D$ — довільні відображення. Необхідно показати, що $(F * F_1) * F_2 = F * (F_1 * F_2)$.

Знайдемо, чому дорівнює права і ліва частини цієї рівності для довільного елемента a із A . Для лівої частини маємо

$$F * (F_1 * F_2)(a) = F * ((F_1 * F_2)(a)) = (F_1 * F_2)(F(a)) = F_2(F_1(F(a))),$$

а для правої частини

$$(F * F_1) * F_2(a) = F_2((F * F_1)(a)) = F_2(F_1(F(a))).$$

Отже, обидві частини рівняння мають один і той же вираз і тому рівні між собою.

Прикладом відображення **на** і взаємно однозначного відображення може служити відображення, яке зв'язує множину A з її фактор-множиною A/R , де R — деяке відношення еквівалентнос-

ті на A . Відображення $F: A \rightarrow A/R$, яке задає для будь-якого елемента a із A той клас розбиття, якому належить a , називається **натуральним відображенням** A на A/R .

Між відношеннями еквівалентності, заданими на деякій множині, і відображеннями цієї множини на інші множини існує тісний зв'язок. Дійсно, якщо $F_1: A \rightarrow B$ — відображення на, то йому відповідає цілком визначене віднесення еквівалентності R на A : якщо $a, b \in A$, то $aRb \Leftrightarrow F_1(a) = F_1(b)$. Ставлячи у відповідність кожному елементу x із B його повний прообраз в A , одержуємо відображення $F_2: B \rightarrow A/R$, основну властивість якого дає така теорема.

Теорема 1.1.7. *Відображення $F_2: B \rightarrow A/R$ є взаємно однозначним відображенням, причому $F_1 * F_2 = F$, де F — натуральне відображення A на A/R .*

Д о в е д е н н я. Визначимо $F_2: B \rightarrow A/R$ так, щоб $F_2(b) = K(b) = \{a \in A \mid F_1(a) = b\}$. Ясно, що коли $b \neq b'$, то $K(b) \neq K(b')$. Значить, відображення F_2 взаємно однозначне. Далі, нехай $F_1(a) = b$, а $F_2(b) = K(b)$. Тоді $a \in K(b)$. Таким чином, добуток $F_1 * F_2$ збігається з натуральним відображенням F .

Теорема доведена.

Якщо $F: A^n \rightarrow B$, то F називають **n -арною функцією** з A в B , а якщо при цьому $B = \{0, 1\}$, то F називається **n -арним предикатом** на множині A , а елементи 0, 1 — відповідно **хібністю** та **істиною**. Якщо предикат F має таку властивість, що для всіх $(a_1, a_2, \dots, a_n) \in A$

$$F(a_1, a_2, \dots, a_n) = 1 \quad (F(a_1, a_2, \dots, a_n) = 0),$$

то предикат F називається **тотожнім (хібним)** на A .

Між відношеннями і предикатами, заданими на одній і тій же множині A , існує взаємно однозначна відповідність. Дійсно, нехай F — n -арний предикат на A . Сукупність тих послідовностей із A^n , для яких $F(a_1, a_2, \dots, a_n) = 1$, є відношенням на A , яке відповідає предикату F . Навпаки, нехай задане будь-яке n -арне відношення $R \subseteq A^n$ на A . Покладаючи

$$F(a_1, a_2, \dots, a_n) = \begin{cases} 1, & \text{якщо } (a_1, a_2, \dots, a_n) \in R, \\ 0, & \text{якщо } (a_1, a_2, \dots, a_n) \notin R, \end{cases}$$

отримуємо n -арний предикат, який відповідає відношенню R . Таким чином, n -арні відношення і n -арні предикати на довільній множині знаходяться у взаємно однозначній відповідності.

Якщо F — n -арна функція із A^n в A , то F ще називають **n -арною операцією** на A . При $n = 0$ функція F називається **нульарною операцією**, значенням якої є фіксований елемент множини A .

Операція F називається **частковою**, якщо F — часткова функція. Нехай $F: A_1 \times A_2 \times \dots \times A_n \Rightarrow B$ і $F^{-1}(b)$ — повний прообраз елемента b в $A_1 \times A_2 \times \dots \times A_n$ при відображення F . Введена вище множина $\text{Dom}(F) = \bigcup_{b \in B} F^{-1}(b)$ називається **областю визначення**

відображення F , а $\text{Im}(F) = \{b \mid F^{-1}(b) \neq \emptyset\}$ — **областю значень**. Якщо $F: A_1 \times A_2 \times \dots \times A_n \Rightarrow B$ і $F_1: A_1 \times A_2 \times \dots \times A_n \Rightarrow B$, то $F = F_1$ тоді і тільки тоді, коли $\text{Dom}(F) = \text{Dom}(F_1)$, $\text{Im}(F) = \text{Im}(F_1)$ і $F(a_1, a_2, \dots, a_n) = F_1(a_1, a_2, \dots, a_n)$ для будь-якого $(a_1, a_2, \dots, a_n) \in A_1 \times A_2 \times \dots \times A_n$.

Надалі позначатимемо предикати, функції, операції малими латинськими літерами. Якщо Ω — деяка множина предикатів, функцій або операцій, то функція $a\omega$: $\Omega \Rightarrow N$, де N — множина натуральних чисел, називається **функцією арності n** , тобто $a\omega = n$, якщо $\omega \in \Omega$ і має арність n .

10. Відношення часткового порядку. Бінарне відношення 0 , визначене на множині A , називається **частковим порядком** на A , якщо для будь-яких a, b, c із A виконуються властивості:

- a1) $a0a \quad (i_A \subseteq 0)$ (рефлексивність);
- a2) $a0b \text{ і } b0c \Rightarrow a0c \quad (0^2 \subseteq 0)$ (транзитивність);
- a3) $a0b \text{ і } b0a \Rightarrow a = b \quad (0 \cap 0^{-1} \subseteq i_A)$ (антисиметричність).

Частковий порядок на множині A , як правило, позначають символом \leq . Якщо $a \leq b$ для деяких $a, b \in A$, то говорять, що a менше або дорівнює b , а також, що a включається в b або дорівнює b . Якщо $a \leq b$ і $a \neq b$, то говорять, що a строго менше b ($a < b$).

Означення 1.1.1. Транзитивне та іррефлексивне відношення називається відношенням строгого порядку. Це відношення позначають $<$.

Означення 1.1.2. Транзитивне та рефлексивне відношення називається відношенням квазіпорядку. Це відношення позначають \preceq .

З кожним відношенням часткового порядку \leq зв'язане відношення строгого порядку $<$ (цей порядок називають строгою частиною відношення \leq):

$$(a < b) \Leftrightarrow a \leq b \text{ і } a \neq b.$$

Навпаки, з кожним відношенням строгого порядку $<$ пов'язане відношення часткового порядку \leq :

$$(a \leq b) \Leftrightarrow a < b \text{ або } a = b.$$

З кожним відношенням квазіпорядку \preceq зв'язані відношення строгого порядку $<$ та еквівалентності \sim :

$$(a < b) \Leftrightarrow a \preceq b \text{ i } \neg(b \preceq a);$$

$$(a \sim b) \Leftrightarrow a \preceq b \text{ i } \neg(b \preceq a).$$

Будь-яке відношення квазіпорядку \preceq , задане на множині A , індукує відношення часткового порядку \leq на фактор-множині A/\sim :

$$[a]_{\sim} \leq [b]_{\sim} \Leftrightarrow a \preceq b.$$

Дійсно, оскільки відношення \preceq транзитивне і рефлексивне, то і відношення \leq теж буде транзитивним і рефлексивним. Покажемо антисиметричність. Відношення $[a]_{\sim} \leq [b]_{\sim}$ та $[b]_{\sim} \leq [a]_{\sim}$ означають, що $a \preceq b$ і $b \preceq a$. Але звідси випливає, що $a \sim b$, тобто що $[a]_{\sim} = [b]_{\sim}$.

Найпростіші властивості частково упорядкованих множин

Теорема 1.1.8 (принцип двоїстості). *Відношення, обернене до відношення часткового порядку, теж буде відношенням часткового порядку.*

Доведення. Нехай 0^{-1} — відношення, обернене до відношення 0:

61) оскільки $i_A \subseteq 0$, то $i_A = i_A^{-1} \subseteq 0^{-1}$;

62) якщо $0 * 0 \subseteq 0$, то за теоремою 1.1.3 $0^{-1} * 0^{-1} = (0 * 0)^{-1} \subseteq 0^{-1}$;

63) якщо $0 \cap 0^{-1} \subseteq i_A$, то $0^{-1} \cap 0 \subseteq i_A \supseteq 0^{-1} \cap (0^{-1})^{-1}$ в силу співвідношення M1 і теореми 1.1.3, що і потрібно було довести.

Відношення часткового порядку 0^{-1} називається **двоїстим** до відношення часткового порядку 0.

Означення 1.1.3. Відношення часткового порядку \leq^{-1} називається **двоїстим** до відношення часткового порядку \leq .

Відношення \leq^{-1} позначається \geq . Таким чином, $a (\leq^{-1}) b$ означає $b \geq a$. Якщо $a \leq b$ або $b \leq a$, то a, b називають елементами, порівнянними відносно порядку \leq .

З принципу двоїстості випливає, що коли в якому-небудь твердженні про частково упорядковану множину замінити частковий порядок на двоїстий до нього порядок, то одержане твердження теж буде справедливе.

Теорема 1.1.9. *Всяка підмножина частково упорядкованої множини теж буде частково упорядкованою множиною.*

Доведення пропонується як проста вправа.

Означення 1.1.4. Елемент x із множини (A, \leq) називається **мінімальним** (максимальним) елементом A , якщо для всякого еле-

мента a із A , що порівнюється з x , справедлива нерівність $x \leq a$ ($x \geq a$).

Означення 1.1.5. Елемент x із A називається *найбільшим (найменшим)*, якщо $\forall a \in A x \geq a$ ($x \leq a$).

Теорема 1.1.10. В кожній частково упорядкованій множині існує не більше одного найменшого (а згідно з принципом двоїстості і найбільшого) елемента.

Д о в е д е н н я. Припустимо, що a і b — два найменші елементи в множині A . Тоді $a \leq b$, оскільки a — найменший елемент, і $b \leq a$, оскільки b — найменший елемент. Але тоді із транзитивності відношення випливає, що $a = b$.

Якщо довільні два елементи з множини A порівнюються між собою відносно \leq , то таке відношення називається *лінійним порядком* на A , а множина A — *упорядкованою* або *лінійно упорядкованою*, або *ланцюгом*.

Ясно, що коли лінійно упорядкована множина A має найбільший (найменший) елемент, то цей елемент буде єдиним максимальним (мінімальним) елементом.

Означення 1.1.6. Нехай (A, \leq) — частково упорядкована множина і $a, b \in A$. Говорять, що елемент b домінує над елементом a , якщо $b > a$ і для жодного елемента x із A не виконується, що $b > x > a$.

Наступна проста теорема показує, що відношення часткового порядку \leq можна однозначно відновити за відношенням домінування в будь-якій скінченній частково упорядкованій множині.

Теорема 1.1.11. Нехай $a < b$ в скінченній частково упорядкованій множині (A, \leq) . Тоді в (A, \leq) існує хоча б один ланцюг $a < x_1 < x_2 < \dots < x_n < b$, в якому кожний x_i домінує над x_{i-1} , $i = 1, 2, \dots, n$.

Д о в е д е н н я індукцією за числом n елементів у, які задовольняють умову $a < u < b$ (див. нижче метод трансформаційної індукції).

При $n = 0$ b домінує над a за означенням.

К р о к і н д у к ц і ї. Припустимо, що теорема справедлива для всіх $m < n$. Розглянемо випадок $n = m$, де $n > 0$. Оскільки $n > 0$, то існує такий елемент $c \in A$, що $a < c < b$, і число елементів y, z , що задовольняють умови $a < y < c$ та $c < z < b$, не перевищує $n - 1$. За припущенням індукції існують скінченні ланцюги, які зв'язують a і c та c і b , середні елементи яких знаходяться у відношенні домінування. З'єднавши ці два ланцюги, одержуємо шуканий ланцюг.

Приклади лінійно упорядкованих множин 1.1.11

1. Множини N , N^+ , Z , RC і множина D дійсних чисел з їх природним порядком; множину N часто називають натуральним рядом, а всяку її підмножину виду $\{0, 1, 2, \dots, n\}$ — початковим відрізком або просто відрізком натурального ряду.

2. Множина точок числової осі (прямої). \blacktriangleleft

Приклади частково упорядкованих множин 1.1.12

3. Булеван $B(A)$, тобто множина всіх підмножин деякої множини A з відношенням теоретико-множинного включення \subseteq як відношенням часткового порядку.

4. Множина N^+ з відношенням $n \leq n_1 \Leftrightarrow n_1$ ділиться націло на n . \blacktriangleleft

Лінійно упорядкована множина A називається *повністю упорядкованою*, якщо всяка її непуста підмножина B має найменший елемент. З поняттям повністю упорядкованої множини пов'язаний один з основних постулатів теорії множин.

Аксіома повної упорядкованості. *Всяку непусту множину можна повністю упорядкувати.*

Зауважимо, що ця аксіома логічно еквівалентна другій аксіомі теорії множин — аксіомі вибору [42, 50, 65].

Аксіома вибору. *Якщо дана множина A , то існує функція f , яка ставить у відповідність кожній непустій підмножині B із множини A один визначений елемент $f(B)$ із множини B .*

Логічну еквівалентність аксіом слід розуміти так: коли одну із них вибрати за аксіому, то другу можна строго довести як теорему, і навпаки. Так, якщо береться аксіома вибору, то аксіома повної упорядкованості стає твердженням, яке в теорії множин відоме як теорема Цермело.

Завдяки аксіомі повної упорядкованості багато властивостей повністю упорядкованих множин можна доводити методом трансфінітної індукції.

Теорема 1.1.12 (метод трансфінітної індукції). *Нехай e — найменший елемент повністю упорядкованої множини A і $P(x)$ — деяка властивість елемента $x \in A$. Тоді, якщо з істинності $P(e)$ і $P(x)$ для всіх $x < e$ випливає істинність $P(a)$, то $P(x)$ істинне для всіх x із A .*

Доведення. Припустимо супротивне: існує така непуста підмножина A' елементів із A , що для всіх $y \in A'$ властивість $P(y)$

хібна при виконанні умов теореми. Нехай b — мінімальний елемент в A' . Оскільки $P(e)$ істинне, то $b \neq e$ і $b > e$. Із умов теореми випливає, що $P(x)$ істинне для всіх $x < b$, але тоді з цих же умов повинна випливати істинність і $P(y)$, а це суперечить нашому припущення.

Теорема доведена.

Метод трансфінітної індукції дає можливість не тільки доводити властивості за індукцією, але і виконувати побудову за індукцією або давати означення за індукцією.

Дійсно, нехай A — повністю упорядкована множина, і нехай потрібно визначити на цій множині функцію $f(x)$, яка ставить у відповідність кожному елементу x із A деякий елемент множини B . Припустимо також, що $f(x)$ повинна задовольняти деякі **рекурентні співвідношення**, тобто співвідношення, які однозначно визначають для всякого $a \in A$ значення $f(a)$ за значеннями $f(b)$ для всіх $b < a$.

Теорема 1.1.13 (метод побудови за індукцією). *Існує єдина функція $f(x)$, яка визначена на всій множині A , задовольняє вказані рекурентні співвідношення і набуває довільних заданих значень на мінімальному елементі множини A .*

Д о в е д е н н я. Покажемо спочатку єдиність такої функції. Припустимо, що існує дві різні функції $f(x)$ і $g(x)$ на множині A , які задовольняють наші умови. Нехай існує непуста підмножина елементів x із A , для яких $f(x) \neq g(x)$. Оскільки A повністю упорядкована, то ця підмножина має мінімальний елемент a . Цей елемент не може бути мінімальним для всієї множини A , тому що тоді, за умовою, на цьому елементі $f(a)$ і $g(a)$ збігалися б. Отже, існує $b < a$, такий, що $f(b) = g(b)$. За умовою теореми рекурентні співвідношення однозначно визначають значення наших функцій для $x = a$ за їх значеннями для всіх $b < a$, отже, $f(a) = g(a)$. Одержані суперечності доводить єдиність $f(x)$.

Доведемо тепер існування функції. Припустимо, що на мінімальному елементі множини A значення шуканої функції уже задано. Позначимо через P таку властивість елемента $a \in A$: a має властивість P , якщо на множині C всіх таких x , що $x \leq a$, може бути визначена функція $f_a(x)$, яка задовольняє рекурентні співвідношення і набуває заданого значення на мінімальному елементі множини A .

За припущенням, P істинне на мінімальному елементі $a \in A$. Далі, якщо елементи b і c задовольняють властивість P і $b < c$, то згідно з доведеною вище єдиністю шуканої функції не на множині A , а на множині $B = \{x \in A \mid x \leq b\}$ маємо $f_b(x) = f_c(x)$.

Звідси випливає, що коли всі елементи b , строго менші елемента a , мають властивість P , то і сам елемент a має цю вла-

стивість. Одержано функцію $f_a(x)$, яка задовольняє всі вимоги, якщо для всякого $b < a$ покласти $f_a(b) = f_b(b)$, а за $f_a(a)$ взяти те значення, яке однозначно визначається за рекурентними співвідношеннями.

На основі методу трансфінітної індукції можна говорити, що для всіх a із A істинне $P(a)$. Покладаючи тепер $\forall a \in A f_a(a) = f(a)$, визначаємо функцію $f(x)$, яка має необхідні властивості.

Теорема доведена.

Зауважимо, що метод трансфінітної індукції, як і метод побудови за індукцією, можна застосовувати і до частково упорядкованих множин. Обмежимося лише формулюваннями цих фактів.

Теорема 1.1.14 (умова індуктивності). *Всі елементи частково упорядкованої множини A мають властивість P , якщо:*

1) *всі мінімальні елементи із множини A задовольняють властивість P (в тому випадку, коли вони існують);*

2) *з того, що $P(x)$ істинне для всіх $x < a$, де $x, a \in A$, випливає істинність $P(a)$.*

Теорема 1.1.15 (побудова за індукцією). *Існує єдина функція $f(x)$, яка визначена на всій множині A , задовольняє вказані рекурентні співвідношення і набуває довільних заданих значень на всіх мінімальних елементах множини A [42], [50].*

5. ПРИКЛАДИ ВІДОБРАЖЕНЬ

На завершення цього підрозділу розглянемо кілька прикладів відображення.

5.1. Потужність множини. Нехай A і B – деякі множини.

Означення 1.1.7. Множини A і B називаються *рівнопотужними*, якщо між їх елементами існує взаємно однозначна відповідність.

Очевидно, що відношення рівнопотужності є відношенням еквівалентності, і тому рівнопотужні множини часто називають *еквівалентними*.

Означення 1.1.8. *Потужністю* або *кардинальним числом* множини A називається клас еквівалентності, якому належить множина A , за відношенням рівнопотужності.

Якщо A і B – скінченні множини, то їх рівнопотужність означає, що вони мають одне і те ж число елементів. При цьому пустій множині \emptyset відповідає число нуль, а скінченній множині, яка складається із n елементів, – число n . Право на таке співставлення нам дає така теорема.

Теорема 1.1.16 (основна теорема про скінченні множини). Всяка скінченна множина не може бути еквівалентна жодній своїй власній надмножині.

Доведення. Припустимо, що B є деяка скінченна надмножина скінченної множини A , яка еквівалентна A , тобто існує взаємно однозначна відповідність $f: A \rightarrow B$. Нехай a_1, \dots, a_n — елементи множини A , а $f(a_1), f(a_2), \dots, f(a_n)$ — їх образи. Серед образів повинні бути всі елементи множини A і хоча б один елемент, який позначимо a_{n+1} .

Для $n = 1$ суперечність очевидна, оскільки єдиний елемент a_1 не може мати два різних образи — a_1, a_2 .

Нехай неможливість існування відображення f із вказаними властивостями доведена для множини A з $n - 1$ елементами. Доведемо її для множини з n елементами.

Можна вважати, що $f(a_n) = a_{n+1}$, бо коли це не так, $f(a_n) = a'$ і $a \neq a_{n+1}$, і значить, a має інший прообраз — $a_i: f(a_i) = a_{n+1}$. Тоді можна побудувати відмінне від f відображення, яке буде ставити у відповідність елементу a_n елемент a_{n+1} , елементу a_i — елемент a' , а всі інші не будуть відрізнятися від f .

Підмножина $A' = \{a_1, a_2, \dots, a_{n-1}\}$ відображається функцією f на деяку множину $f(A')$, яка будеться із $f(A) = B$ за допомогою відкидання елемента $f(a_n) = a_{n+1}$.

Множина $f(A')$ містить елементи a_1, \dots, a_n і, значить, є власною надмножиною множини A і водночас її взаємно однозначним образом. Але за припущенням індукції це неможливо.

Теорема доведена.

З цієї теореми випливає, що скінченна множина ніколи не може бути еквівалентна двом різним відрізкам натурального ряду, тому що в протилежному випадку ці відрізки були б рівнопотужними і при цьому один із них мав би містити другий. Таким чином, всяка скінченна множина A еквівалентна одному і тільки одному відрізку $(0, 1, 2, \dots, n)$ натурального ряду. Однозначно визначене число m — число елементів множини A — може слугувати мірою потужності цієї множини.

Теорема 1.1.17. Якщо A_1, A_2, \dots, A_n — скінченні множини, то

$$|A_1 \times A_2 \times \dots \times A_n| = |A_1| |A_2| \dots |A_n|.$$

Доведення. Спочатку покажемо, що $|A_1 \times A_2| = |A_1| |A_2|$. Нехай $A_1 = \{a_{11}, a_{12}, \dots, a_{1n}\}$, $A_2 = \{a_{21}, a_{22}, \dots, a_{2k}\}$ і $B(a_{1j}) = \{(a_{1i}, a_{2j})\}$, де a_{1i} — деякий фіксований елемент множини A_1 . Тоді $|A_2| = |B(a_{1j})|$, оскільки ці множини еквівалентні (елементу (a_{1i}, a_{2j}) відповідає елемент a_{2j}). Звідси маємо

$$|A_1 \times A_2| = |B(a_{11})| + |B(a_{12})| + \dots + |B(a_{1n})| = |A_1| |A_2|,$$

тому що множини $B(a_{1i})$ і $B(a_{lj})$ попарно не перетинаються при $i \neq j$.

Звертаючи увагу на те, що множини $A_1 \times (A_2 \times \dots \times A_n)$ і $A_1 \times A_2 \times \dots \times A_n$ еквівалентні (елементу $[a_1, (a_2, \dots, a_n)]$ відповідає елемент (a_1, a_2, \dots, a_n)), можемо записати:

$$\begin{aligned} |A_1 \times A_2 \times \dots \times A_n| &= |A_1| (|A_2 \times \dots \times A_n|) = \\ &= |A_1| |A_2| (|A_3 \times \dots \times A_n|) = \dots = |A_1| |A_2| \dots |A_n|. \end{aligned}$$

Теорема доведена.

Множина, еквівалентна множині натуральних чисел, називається **зліченою** множиною.

Безпосередньо з означення зліченої множини випливає таке твердження.

Твердження 1.1.1. *Всяка нескінчenna підмножина зліченої множини сама злічена.*

Дійсно, перерахунок елементів підмножини B зліченої множини A можна виконати в порядку їх слідування в множині A .

У випадку нескінчених множин аналог теореми 1.1.16 не має місця, як показує така теорема.

Теорема 1.1.18. *Об'єднання скінченої або зліченої множини злічених множин знову є множиною зліченою* [19].

Д о в е д е н н я. Розглянемо спочатку випадок скінченного числа злічених множин. Нехай A_1, \dots, A_k — злічені множини і $a_{ij} \in A_i$ — їх елементи. Розглянемо послідовність

$$a_{11}, \dots, a_{k1}, \dots, a_{12}, \dots, a_{k2}, \dots, a_{1n}, \dots, a_{kn}, \dots$$

Таку послідовність можна перерахувати, і якщо деякий елемент при переліку вже зустріався раніше і одержав номер, то далі він пропускатиметься. Отже, множина $A = \bigcup_{i=1}^k A_i$ злічена.

Нехай маємо зліченну множину злічених множин $A_1, A_2, \dots, A_n, \dots$, де $A_i = \{a_{i1}, a_{i2}, \dots\}$. Існує лише скінченне число елементів a_{ik} , для яких $i + k = 2$, аналогічно існує лише скінченне число елементів a_{ik} , для яких $i + k = 3$, і т.д. Перенумеруємо спочатку всі елементи, для яких $i + k = 2$ (наприклад, за зростанням значення i), а потім (за допомогою інших чисел) — елементи, для яких $i + k = 3$ і т.д. При цьому кожний елемент a_{ik} одержить деякий номер, і різні елементи будуть мати різні номери. Звідси випливає справедливість теореми.

Наслідок 1.1.1. Множина \mathbf{Z} всіх цілих чисел злічена.

Дійсно, $\mathbf{Z} = \mathbf{N} \cup \mathbf{N}'$, де $\mathbf{N}' = \{-1, -2, \dots, -n, \dots\}$.

Наслідок 1.1.2. Множина \mathbf{RC} всіх раціональних чисел злічена.

Множина раціональних чисел — це об'єднання зліченої сукупності зліченних множин

$$\text{RC} = \mathbf{Z} \cup R_2 \cup R_3 \cup \dots \cup R_n \cup \dots,$$

де $R_n = \{m/n \mid m \in \mathbf{Z}, n = 2, 3, \dots\}$.

Наслідок 1.1.3. Декартовий добуток $A \times B$ зліченних множин є множина зліченна.

Дійсно, якщо $A = \{a_1, \dots, a_n, \dots\}$ і $B = \{b_1, \dots, b_n, \dots\}$, то множину всіх елементів множини $A \times B$ можна представити як об'єднання зліченої сукупності зліченних множин, що мають вигляд

$$\begin{aligned} A_1 &= \{(a_1, b_1), (a_1, b_2), \dots, (a_1, b_n), \dots\}, \\ A_2 &= \{(a_2, b_1), (a_2, b_2), \dots, (a_2, b_n), \dots\}, \\ &\dots \\ A_n &= \{(a_n, b_1), (a_n, b_2), \dots, (a_n, b_n), \dots\}, \\ &\dots \end{aligned}$$

Згідно з теоремою 1.1.18 робимо висновок, що множина $A \times B$ зліченна.

Наслідок 1.1.4. Декартовий добуток $A_1 \times A_2 \times \dots \times A_n$ зліченних множин A_1, A_2, \dots, A_n є множина зліченна для будь-якого n .

Далі будемо мати справу або зі скінченими або нескінченими зліченними множинами. Але для повноти зауважимо, що існують множини, елементи яких перерахувати неможливо. Такі множини логічно називати **незліченними**. Існування таких множин показує наступна теорема.

Теорема Кантора. *Множина всіх дійсних чисел з інтервалу $(0, 1)$ незліченна.*

Доведення цієї теореми базується на **діагональному методі Кантора** [19].

З курсу шкільної математики відомо, що кожному дійсному числу з інтервалу $(0, 1)$ можна однозначно поставити у відповідність правильний нескінчений десятковий дріб $0.a_1a_2\dots a_n\dots$, який має нескінченно багато відмінних від нуля цифр. (Якщо число — скінчений десятковий дріб, наприклад $0,243$, то йому ставиться у відповідність нескінчений десятковий дріб виду $0,24299\dots 99\dots$)

Припустимо, що теорема хибна і множина дійсних чисел з інтервалу $(0, 1)$ зліченна. Тоді для цієї множини повинен бути перелік

$$\begin{aligned} x_1 &= 0, a_{11} a_{12} \dots a_{1n} \dots, \\ x_2 &= 0, a_{21} a_{22} \dots a_{2n} \dots, \\ &\dots \\ x_n &= 0, a_{n1} a_{n2} \dots a_{nn} \dots, \\ &\dots \end{aligned}$$

Прямуючи діагоналлю (тобто елементами $a_{11}, a_{22}, \dots, a_{nn}, \dots$), складемо новий дріб $a_{11}' a_{22}' \dots a_{nn}' \dots$, такий, що $a_{ii} \neq a_{ii}'$ для будь-якого $i = 1, 2, \dots$. Побудований дріб не входить до розглянутого переліку, оскільки відрізняється від будь-якого елемента x_i числом a_{ii} , яке лежить на діагоналі. Отже, переліку для всіх чисел з інтервалу $(0, 1)$ не існує, і припущення про зліченність цієї множини хибне.

Теорема доведена.

Наслідок 1.1.5. Якщо A — незліченна множина і $B \subset A$ — деяка зліченна підмножина множини A , то множина $C = A \setminus B$ незліченна.

Дійсно, якби множина C була зліченою, то зліченою була б і множина $A = C \cup B$ за теоремою 1.1.18.

Наслідок 1.1.6. Множина \mathbf{IR} всіх ірраціональних чисел незліченна.

Множина, еквівалентна множині дійсних чисел з інтервалу $(0, 1)$, називається **множиною потужності континууму**.

Теорема про булеан. Булеан $\mathbf{B}(U)$ деякої зліченої множини U — незліченна множина.

Д о в е д е н н я. Не обмежуючи загальності, візьмемо $U = \mathbf{N}$. Поставимо у відповідність кожній підмножині $N_j \subset \mathbf{N}$ ланцюжок, який складається з нулів і одиниць таким чином: елемент a_{ij} , який стоїть на i -му місці в множині N_j , дорівнює нулю, якщо $a_{ij} \in N_j$, і 1, якщо $a_{ij} \notin N_j$. Очевидно, що така відповідність буде взаємно однозначною. Використовуючи діагональний метод Кантора, побудуємо новий ланцюжок:

$$a_{11}' a_{22}' \dots a_{nn}' \dots,$$

де

$$a'_{ii} = \begin{cases} 1, & \text{якщо } a_{ii} = 0, \\ 0, & \text{якщо } a_{ii} \neq 0, \end{cases}$$

$i = 1, 2, \dots, n, \dots$ Ясно, що такий ланцюжок не збігається ні з яким ланцюжком для множини N_j і відповідає деякій підмножині $N' \subset \mathbf{N}$, яка не збігається ні з однією N_j , $j = 1, 2, \dots, n, \dots$ Отже, елементи $\mathbf{B}(U)$ неможливо перелічити.

Теорема доведена.

5.2. Матриці. Розглянемо ще один цікавий приклад відображення. Нехай $p, q \in \mathbf{N}^+$, N_p і N_q означають множини чисел $\{1, 2, \dots, p\}$ і $\{1, 2, \dots, q\}$ відповідно, а P — довільна множина чисел.

Відображення $A: N_p \times N_q \rightarrow P$, де $p, q \in \mathbf{N}^+$, називається прямокутною **матрицею** над множиною чисел P . Образ $A(i, j)$ часто позначають a_{ij} і називають елементом матриці A , а саму матрицю

A задають за допомогою таблиці елементів-образів із множини P , які відповідають відображення A , тобто

$$A = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1q} \\ a_{21} & a_{22} & \dots & a_{2q} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ a_{p1} & a_{p2} & \dots & a_{pq} \end{vmatrix}.$$

У цьому випадку говорять, що матриця A складається з p рядків і q стовпчиків і має розмірність $p \times q$. Якщо $p = q$, то матриця A називається **квадратною**, а коли елементи квадратної матриці такі, що $a_{ij} = 0$ при $i \neq j$, і $a_{ii} \neq 0$ при $i = j$, то матриця A називається **діагональною**. Діагональна матриця називається **одиничною**, якщо $a_{ii} = 1$ для всіх i .

Матриця являє собою один із зручних засобів задання бінарних відношень на скінчених множинах.

Нехай R — бінарне відношення, задане на скінчених множинах $A = \{a_1, a_2, \dots, a_p\}$ і $B = \{b_1, b_2, \dots, b_q\}$. Розглянемо матрицю $A(R)$: $N_p \times N_q \Rightarrow \{0, 1\}$, яка зв'язана з відношенням R (за допомогою відображення A) таким чином:

$$a'_{ii} = \begin{cases} 1, & \text{якщо } (a_i, b_j) \in R, \\ 0, & \text{якщо } (a_i, b_j) \notin R. \end{cases}$$

Наприклад, якщо $A = \{a_1, a_2, a_3, a_4\}$, $B = \{b_1, b_2, b_3\}$, $R = \{(a_1, b_2), (a_1, b_3), (a_2, b_1), (a_3, b_1), (a_4, b_2)\}$, то матриця $A(R)$ має такий вигляд:

$$A(R) = \begin{vmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{vmatrix}.$$

Зауважимо, що в окремому випадку, коли R — бінарне відношення, задане на скінченній множині A , то:

а) якщо $R = i$ — відношення тотожності на A , то

$$A(i) = \begin{vmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{vmatrix}$$

є одинична діагональна матриця (звідки і назва відношення). Звідси легко знайти рефлексивне замикання деякого заданого відношення R на множині A . Для цього в матриці $A(R)$ необхідно скрізь поставити на діагоналі 1 замість 0;

б) якщо R — симетричне відношення, то матриця $A(R)$ буде, очевидно, симетричною. Дійсно, в силу симетричності відношення R маємо: якщо $(a_i, b_j) \in R$ ($a_{ij} = 1$), то $(b_j, a_i) \in R$ ($a_{ji} = 1$). Беручи до уваги цю обставину, легко побудувати матрицю симетричного замикання $A(R_s)$ при умові, що $A(R)$ є матриця деякого відношення R . Для цього необхідно в $A(R)$ замінити 0 на 1 на відповідних місцях з урахуванням симетрії.

Наприклад, якщо $A = \{a_1, a_2, a_3, a_4\}$, $R = \{(a_1, a_3), (a_1, a_4), (a_2, a_2), (a_2, a_4), (a_3, a_4)\}$ і

$$A(R) = \begin{vmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{vmatrix}, \quad \text{то} \quad A(R_s) = \begin{vmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{vmatrix}.$$

Більше того, якщо відношення R має k елементів, серед яких r ($r < |A|$) діагональних, то R_s має $2(k - r) + r = 2k - r$ елементів. Отже, коли $A(R_s)$ має непарне число елементів, які дорівнюють одиниці, то R_s повинне включати хоча б одну діагональну пару.

ДОДАТОК. Аксіоматика Цермело—Френкеля

Наведемо для повноти повний перелік аксіом теорії множин, який має назву **аксіоматика Цермело—Френкеля** [32].

- **Аксіома об'ємності.** Дві множини рівні тоді і тільки тоді, коли вони складаються з одних і тих же елементів.
- **Аксіома згортки.** Будь-яка властивість $P(x)$ визначає деяку множину A за допомогою умови: елементами множини A є ті і тільки ті предмети a , які мають властивість P .
- **Аксіома пари.** Якщо a і b — різні предмети, то існує множина $\{a, b\}$, яка складається точно з предметів a і b .
- **Аксіома об'єднання.** Для довільної множини A існує множина B , яка складається точно з усіх елементів, що входять до елементів множини A .
- **Аксіома нескінченості.** Існує хоча б одна нескінчена множина — множина натуральних чисел N .
- **Аксіома булеана.** Для будь-якої множини A існує множина $B(A)$ всіх підмножин множини A .

• **Аксіома вибору.** Якщо дана множина A , то існує функція f , яка ставить у відповідність кожній непустій підмножині B із множини A один певний елемент $f(B)$ із множини B .

• **Аксіома підстановки.** Для всякої множини A і однозначної функції f , визначененої на множині A , існує множина, яка складається точно з елементів $f(x)$ для $x \in A$.

Наведена система аксіом є недостатньо визначеною. Це пов'язано з питанням висловлювання, яке використовується в аксіомі згортки. Мова висловлювань повинна описуватись точніше, інакше це призведе до суперечностей, які відомі як парадокси теорії множин типу: "я кажу неправду" (*парадокс брехуна*), "Нехай A — множина всіх множин, що не є елементами самих себе, тоді кожне з двох можливих висловлювань є суперечним: A є елементом A і A не є елементом A " (*парадокс Рассела*) і т. п.

Уточнення формальної мови висловлювань детально розглянуто нижче.

Контрольні питання

1. Що таке множина?
2. Які існують способи задання множин?
3. Які основні операції виконуються над множинами?
4. Що таке булеван деякої множини U ?
5. Що таке:
 - а) декартовий добуток множин A, B, C, E ;
 - б) декартова степінь деякої множини A ;
 - в) бінарне відношення, задане на множині A ?
6. Назвіть основні властивості бінарних відношень. Яке відношення називається рефлексивним, симетричним, антисиметричним, транзитивним?
7. Чи будуть відношеннями еквівалентності відношень $R_1 = R \cap T$, $R_1 = R * T$ і $R_1 = R^{-1}$, якщо R і T — відношення еквівалентності?
8. Чи буде бінарне відношення R таке, що $(a, b) R (c, d) \Leftrightarrow a + d = b + c$, де $a, b, c, d \in \mathbb{N}^+$, відношенням еквівалентності?
9. Чи буде відношенням часткового порядку відношення, обернене до відношення часткового порядку?
10. Які властивості матиме матриця рефлексивного, симетричного, антисиметричного відношенні, заданого на деякій непустій скінченний множині?
11. Чи будуть еквівалентними множини $\{a, b, c, d\}$ і $\{a, b, c, d, d\}$?

Доведіть, що множини парних і непарних натуральних чисел еквівалентні.

Задачі та вправи

1. Довести, що умови

$$\begin{aligned}A &\subseteq B, \\A \cap B &= A, \\A \cup B &= B\end{aligned}$$

еквівалентні між собою.

2. Довести, що

$$\begin{aligned}A \cap B &\subseteq A \subseteq A \cup B, \\A \cap B &\subseteq B \subseteq A \cup B, \\A \setminus B &\subseteq A.\end{aligned}$$

3. Довести, що

$$\begin{aligned}\emptyset &\neq \{\emptyset\}, \\\{\{a\}, \{b, c\}\} &\neq \{a, b, c\}.\end{aligned}$$

4. Пояснити, чому

$$\begin{aligned}3 &\in \{1, 2, 3, 4\}, \\\{1, 2\} &\notin \{\{1, 2, 3\}, \{2, 3\}, 1, 2\}.\end{aligned}$$

5. Описати словами кожну з множин:

- $\{x \in \mathbb{N} \mid x \text{ ділиться на } 2 \text{ і } x \text{ ділиться на } 3\};$
- $\{x \mid x \in A \text{ і } x \in B\};$
- $\{x \mid x \in A \text{ і } x \notin B\};$
- $\{(x, y) \in \mathbf{D}^2 \mid x^2 + y^2 = 1\};$
- $\{(x, y) \in \mathbf{D}^2 \mid y = 2x \text{ і } y = 3x\}.$

6. Нехай універсальною множиною U служить множина натуральних чисел \mathbb{N} , тобто $U = \mathbb{N}$, а

$$\begin{aligned}A &= \{x \in \mathbb{N} \mid \text{для деякого } y \in \mathbb{N}^+ x = 2y\}, \\B &= \{x \in \mathbb{N} \mid \text{для деякого } y \in \mathbb{N}^+ x = 2y - 1\}, \\C &= \{x \in \mathbb{N} \mid x < 10\}.\end{aligned}$$

Побудувати або описати словами множини

$$A', (A \cup B)', C', A \setminus C', C' \setminus (A \cup B).$$

7. Знайти множини:

$$\begin{aligned}\emptyset \cap \{\emptyset\}, \\\{\emptyset\} \cap \{\emptyset\}, \\\{\emptyset, \{\emptyset\}\} \setminus \emptyset, \\\{\emptyset, \{\emptyset\}\} \setminus \{\emptyset\}, \\\{\emptyset, \{\emptyset\}\} \setminus \{\{\emptyset\}\}.\end{aligned}$$

8. Довести, що множина всіх коренів многочлена $F(x) = f_1(x) \cdot f_2(x)$ — це об'єднання множин коренів многочленів $f_1(x)$ і $f_2(x)$.

9. Довести:

- закони М1 — М5 — алгебри множин;
- $((A \cap B \cap X) \cup (A \cap B \cap C \cap X \cap Y) \cup (A \cap X \cap X')) =$
 $= A \cap B \cap X.$

10. Довести тотожності:

$$\begin{aligned}A \cup A' &= U; \\A \setminus (B \cup C) &= (A \setminus B) \cap (A \setminus C); \\A \setminus (B \cup C) &= (A \setminus B) \setminus C;\end{aligned}$$

$$\begin{aligned}
A \setminus (B \cap C) &= (A \setminus B) \cup (A \setminus C); \\
A \setminus (B \setminus C) &= (A \setminus B) \cup (A \cap C); \\
(A \setminus B) \setminus C &= (A \setminus C) \setminus (B \setminus C); \\
A \cap B &= A \setminus (A \setminus B); \\
A \cup B &= A \cup (B \setminus A); \\
(A' \cup B) \cap A &= A \cap B; \\
A \cap (B \setminus A) &= \emptyset; \\
(A \cup B) \setminus C &= (A \setminus C) \cup (B \setminus C);
\end{aligned}$$

Довести співвідношення:

$$\begin{aligned}
A \cup B \subseteq C &\Leftrightarrow A \subseteq C \text{ i } B \subseteq C; \\
A \subseteq (B \cap C) &\Leftrightarrow A \subseteq B \text{ i } A \subseteq C; \\
A \cap (B \setminus C) &= (A \cap B) \setminus (A \cap C) = (A \cap B) \setminus C; \\
(A \cap B) \cup (A \cap B') &= (A \cup B) \cap (A \cup B') = A; \\
A = B &\Leftrightarrow A \cup B = B \Leftrightarrow A \cap B = A \Leftrightarrow A \setminus B = \emptyset \text{ i } A' \cup B = U,
\end{aligned}$$

де U — універсальна множина, а A' — доповнення множини A в U .

11. Чи існують такі множини A , B , C , що

$$A \cap B \neq \emptyset, A \cap C = \emptyset, (A \cap B) \setminus C = \emptyset?$$

12. Знайти всі підмножини множин: \emptyset , $\{\emptyset\}$, $\{x\}$, $\{1, 2\}$.

13. Довести, що множина A , яка складається із n елементів, має 2^n підмножин.

14. Що являє собою множина $\mathbf{D} \times \mathbf{D}$, якщо \mathbf{D} — множина дійсних чисел?

15. Довести, що $A \subseteq (B \cup C) \Leftrightarrow (A \cap B') \subseteq C$.

16. Які з наведених нижче тверджень справедливі для будь-яких множин A , B , C :

$$A \subseteq B \text{ i } B \subseteq C \Rightarrow A \subseteq C;$$

$$A \neq B \text{ i } B \neq C \Rightarrow A \neq C;$$

$$A \subseteq (B \cup C)' \text{ i } B \subseteq (A \cup C)' \Rightarrow B = \emptyset.$$

17. Нехай U — універсальна множина і $A \subseteq U$. Довести, що $U = A \Rightarrow \Rightarrow A = U$.

18. Довести, що $A = B' \Leftrightarrow A \cap B = \emptyset \text{ i } A \cup B = U$.

19. Довести, що коли $A_1 \supseteq A_2 \supseteq \dots \supseteq A_n \supseteq A_1$, то $A_1 = A_2 = \dots = A_n$ для довільних множин A_1, A_2, \dots, A_n .

20. Довести, що:

$$A + B = B + A;$$

$$A + (B + C) = (A + B) + C;$$

$$A = B \Leftrightarrow A + B = \emptyset;$$

$$A \cap (B + C) = (A \cap B) + (A \cap C);$$

$$A + (A + B) = B;$$

$$A \cup B = A + B + (A \cap B);$$

$$A \setminus B = A + (A \cap B);$$

$$A + \emptyset = A;$$

$$A + A = \emptyset;$$

$$A + U = A'.$$

21. Довести, що:

$$A \cup B = A \cap B \Rightarrow A = B;$$

$$(A \cap B) \cup C = A \cap (B \cup C) \Leftrightarrow C = A;$$

$$A \subseteq B \Rightarrow A \cup C \subseteq B \cup C;$$

$$A \subseteq B \Rightarrow A \cap C \subseteq B \cap C;$$

$$A \subseteq B \Rightarrow A \setminus C \subseteq B \setminus C;$$

$$A \subseteq B \Rightarrow C \setminus B \subseteq C \setminus A;$$

$$A \subseteq B \Rightarrow B' \subseteq A';$$

$$(A \cap B) \cup C = A \cap (B \cup C) \Leftrightarrow C \subseteq A.$$

22. Довести, що коли A, B, C, D — непусті множини, то:

$$A = B \text{ і } C = D \Leftrightarrow A \times C = B \times D;$$

$$A \subseteq B \text{ і } C \subseteq D \Leftrightarrow A \times C \subseteq B \times D.$$

23. Довести, що:

$$\mathbf{B}(A \cap C) = \mathbf{B}(A) \cap \mathbf{B}(C);$$

$$\mathbf{B}(\cap A_i) = \cap \mathbf{B}(A_i);$$

$$\mathbf{B}(\cup A_i) = \{ \cup C_i \mid C_i \in \mathbf{B}(A_i) \},$$

де i пробігає деяку множину цілих чисел I .

24. Довести, що $\bigcap_{i=1}^n (B \cup A_i) = B \cap (\bigcup_{i=1}^n A_i)$.

25. Довести, що $\{\{x\}, \{x, y\}\} = \{\{z\}, \{z, u\}\} \Leftrightarrow x = z \text{ і } y = u$.

26. Довести, що

$$A \times (B \times C) \neq (A \times B) \times C, (A \cap B) \times (C \cap D) = (A \times C) \cap (B \times D).$$

27. Дати геометричну інтерпретацію відношень:

$$\{(x, y) \in \mathbf{D}^2 \mid y = x\};$$

$$\{(x, y) \in \mathbf{D}^2 \mid y \geq x\};$$

$$\{(x, y) \in \mathbf{D}^2 \mid 0 \leq x \leq 1 \text{ або } 0 \leq y \leq 1\};$$

$$\{(x, y) \in \mathbf{D}^2 \mid 0 \leq x \leq 1 \text{ і } 0 \leq y \leq 1\}.$$

28. Знайти R^2 і R^{-1} для відношення $R = \{(x, y) \mid x, y \in \mathbf{N}^+ \text{ і } x \text{ ділить } y\}$.

29. Довести, що для будь-яких бінарних відношень мають місце рівності:

$$(R \cup R_1)^{-1} = R^{-1} \cup R_1^{-1},$$

$$(R')^{-1} = (R^{-1})'.$$

Довести, що коли бінарне відношення R на A :

а) симетричне, то $R = R^{-1}$;

б) рефлексивне і транзитивне, то $R^2 = R$;

в) рефлексивне і антисиметричне, то $R \cap R^{-1} = \emptyset$.

30. Навести приклад бінарного відношення, яке:

а) рефлексивне, симетричне, нетранзитивне;

б) рефлексивне, несиметричне, транзитивне;

в) рефлексивне, антисиметричне, нетранзитивне;

г) нерефлексивне, симетричне, транзитивне.

31. Нехай $A = \{a, b, c, d, e\}$, а $R = \{(a, b), (a, c), (b, b), (c, c), (e, e), (d, d), (d, a), (d, b), (c, d), (e, d)\}$ і $R_1 = \{(a, d), (d, a), (c, d), (c, c), (b, b)\}$ — бінарні відношення на множині A .

Побудувати R^{-1} , $R \cap R_1$, $R \cup R_1$, $R \setminus R_1$, $R \div R_1$.

Чи буде відношення R , $R \cap R_1$, $R \div R_1$, $R \cup R_1$, $R \setminus R_1$ рефлексивним, симетричним, транзитивним?

Побудувати матриці відношень для R , R_1 , $R \cap R_1$, $R \cup R_1$, $R \setminus R_1$, $R \div R_1$, $R_1 \cup R_2$, R^2 , R^3 .

32. Яке відношення є транзитивним замиканням відношення “ X — прямий нащадок Y ”?

33. Нехай $A = \{2, 3, 5, 6, 11, 12, 14\}$ та \leq — відношення часткового порядку на A , таке, що $x \leq y \Leftrightarrow x$ ділить y .

Побудувати множину відношення \leq та упорядкувати множину A відносно цього порядку.

34. Показати, що відношення \subseteq для множин є відношенням часткового порядку.

Для яких множин A булеван $\mathbf{B}(A)$ буде лінійно упорядкованою множиною відносно відношення \subseteq ?

35. Довести, що коли f — функція із A в B , а g — функція із B в C , то $f * g$ є функцією із A в C .

36. Довести, що для того щоб відображення $R: A \rightarrow B$ було взаємно однозначним, необхідно і достатньо, щоб $i_A = R * R^{-1}$, $R^{-1} * R = i_B$.

37. Довести, що коли f — довільна функція, то $f(A \cap B) \subseteq f(A) \cap f(B)$, ($A \subseteq B \Rightarrow f(A) \subseteq f(B)$).

38. Встановити взаємно однозначну відповідність між множинами $A \times B$ і $B \times A$.

39. Довести наслідок 1.1.4.

40. Довести наслідок 1.1.6.

41. Довести, що множина дійсних чисел незліченна.

42. Довести методом математичної індукції, що:

$$n^5 - n \text{ кратне } 5;$$

$$n \cdot (2n^2 - 3n + 1) \text{ кратне } 6;$$

$$n^7 - n \text{ кратне } 7;$$

$$5 \cdot 2^{3n-2} + 3^{3n-1} \text{ кратне } 19;$$

$$1^3 + 2^3 + \dots + n^3 = (n \cdot (n+1)/2)^2;$$

$$1^2 + 2^2 + \dots + (2n-1)^2 = n \cdot (4n^2 - 1)/3;$$

$$(1 - 1/4) \cdot (1 - 1/9) \cdot \dots \cdot (1 - 1/(n+1)^2) = (n+2)/(2n+2);$$

$$1/(n+1) + 1/(n+2) + \dots + 1/2n > (13/24);$$

$$1 \cdot 2 \cdot 2 \cdot 5 + \dots + n \cdot (3n-1) = n^2 \cdot (n+1);$$

$$1/2 \cdot 3/4 \cdot 5/6 \cdot \dots \cdot (2n-1)/n < 1/\sqrt{3n+1}.$$

43. Довести, що коли x_1, x_2, \dots, x_n — додатні дійсні числа, такі, що $x_1 \cdot x_2 \cdot \dots \cdot x_n = 1$, то $x_1 + x_2 + \dots + x_n \geq n$.

Користуючись доведеною нерівністю, довести нерівність

$$\frac{x_1+x_2+\dots+x_n}{n} \geq \sqrt{x_1x_2\dots x_n}.$$

44. Довести, що коли x_1, x_2, \dots, x_n — додатні дійсні числа, такі, що $x_1 + x_2 + \dots + x_n \leq 1/2$, то

$$(1-x_1) \cdot (1-x_2) \cdot \dots \cdot (1-x_n) \geq 1/2.$$

45. Довести, що довільну суму грошей, більшу 7 гривень, можна розміняти купюрами вартістю 3 і 5 гривень.

§ 1.2. ЕЛЕМЕНТИ КОМБІНАТОРНОГО АНАЛІЗУ

На практиці люди часто зустрічаються з задачами, в яких необхідно підрахувати число всіх можливих способів розміщення деяких предметів скінченної множини або число всіх можливих способів виконання певної дії із скінченної множини таких дій. Наприклад, скількома різними способами можна розставити дужки у виразі $a + b + c + d + e + f$, якщо операція $+$ асоціативна, а букви a, b, c, d, e, f — деякі дійсні числа? Скількома способами можуть розподілитися медалі на чемпіонаті світу з футболу серед 16-ти команд-учасниць фінальної групи? Задачі такого типу називаються *комбінаторними*, а методи їх розв'язку — *методами комбінаторного аналізу*. Оскільки комбінаторика має справу із скінченими множинами, то її часто називають *теорією скінчених множин*.

1. ОСНОВНЕ ПРАВИЛО КОМБІНАТОРИКИ

Розглянемо таку задачу. Нехай з пункту A міста Києва в пункт B можна доїхати трьома видами транспорту: тролейбусом (T), автобусом (A) і метро (M), а з пункту B в пункт C — лише двома видами транспорту: тролейбусом (T) і автобусом (A). Скількома способами можна доїхати з пункту A в пункт C міста Києва? Розв'язок цієї задачі зводиться, очевидно, до підрахунку числа елементів в декартовому добутку множин $\{A, T, M\} \times \{A, T\}$. Число таких елементів, як ми знаємо, дорівнює добутку числа елементів першої множини на число елементів другої множини, тобто в нашому випадку це $3 \cdot 2 = 6$. Отже, існує шість способів доїхати із пункту A в пункт C . Виявляється, що за цією простою задачею ст縟ить правило, яке називається *основним правилом комбінаторики*.

Нехай необхідно виконати послідовно k дій. Якщо першу дію можна виконати n_1 способами, другу — n_2 способами і так далі до k -ї дії, яку можна виконати n_k способами, то всі k дій можна виконати $n_1 \cdot n_2 \cdot \dots \cdot n_k$ способами.

Справедливість цього правила безпосередньо випливає з теореми 1.1.17.

Приклади 1.2.1

1. Скількома способами на першості світу з футболу можуть розподілитися медалі, якщо у фінальній частині грають 24 команди?

Р о з в ' я з о к. Золоту медаль може одержати будь-яка з 24-х команд, тобто маємо 24 можливості. Срібну медаль може виграти одна з 23-х команд, а бронзову — одна з 22-х команд. За основним правилом комбінаторики загальне число способів розподілу медалей $24 \cdot 23 \cdot 22 = 12144$.

2. Скільки тризначних чисел можна скласти з цифр 0, 1, 2, 3, 4, 5, якщо:

- 1) цифри можуть повторюватися;
- 2) ні одна з цифр не повторюється двічі;
- 3) цифри непарні і можуть повторюватися.

Р о з в ' я з о к. 1. Першою цифрою може бути одна із цифр 1, 2, 3, 4, 5, оскільки 0 не може бути першою цифрою, бо в такому випадку число не буде тризначним. Якщо перша цифра вибрана, то друга може бути вибрана шістьма способами, як і третя цифра. Отже, загальне число тризначних цифр $5 \cdot 6 \cdot 6 = 180$.

2. Першою цифрою може бути одна з п'яти цифр — 1, 2, 3, 4, 5, якщо перша цифра вибрана, то другою може бути теж одна з п'яти цифр (тут уже враховується 0), а третя може бути вибрана чотирма способами з чотирьох цифр, що залишилися. Отже, загальна кількість таких тризначних чисел $5 \cdot 5 \cdot 4 = 100$.

3. Першою цифрою може бути одна з трьох цифр: 1, 3, 5. Другою теж може бути одна з цих трьох цифр. Аналогічно і третя цифра може бути вибрана трьома способами. Таким чином, загальна кількість таких чисел дорівнює $3 \cdot 3 \cdot 3 = 27$. \blacktriangleleft

2. ЧИСЛО РІЗНИХ k -ЕЛЕМЕНТНИХ ПІДМНОЖИН n -ЕЛЕМЕНТНОЇ МНОЖИНИ

Подивимось тепер, скільки існує різних підмножин із k елементів в множині, що має n елементів ($k < n$).

Теорема 1.2.1. Число різних k -елементних підмножин n -елементної множини становить

$$C_n^k = \frac{n!}{k!(n-k)!}, \quad (1.2.1)$$

де скорочення $n! = n \cdot (n-1) \cdots 3 \cdot 2 \cdot 1$ називається **факторіалом числа n** (читається n -факторіал).

Д о в е д е н н я. Щоб побудувати k -елементну підмножину множини A , необхідно до $(k-1)$ -елементної множини приєднати один із $n-k+1$ елементів, які не входять в цю підмножину. Оскільки число $(k-1)$ -елементних підмножин дорівнює C_n^{k-1} і

кожну з цих підмножин можна зробити k -елементною $n - k + 1$ способами, то згідно з основним правилом комбінаторики одержуємо число $C_n^{k-1} \cdot (n - k + 1)$ підмножин. Але не всі ці підмножини будуть різними, оскільки всяку k -елементну множину можна так побудувати k способами. Отже,

$$\begin{aligned} C_n^k &= \frac{n - k + 1}{k} C_n^{k-1} = \frac{(n - k + 1)(n - k + 2)}{k \cdot (k - 1)} C_n^{k-2} = \dots = \\ &= \frac{(n - k + 1)(n - k + 2) \dots (n - 1)}{k \cdot (k - 1) \cdot \dots \cdot 2} C_n^1. \end{aligned}$$

Оскільки C_n^1 — число одноелементних підмножин множини A дорівнює n , то

$$C_n^k = \frac{n(n - 1) \dots (n - k + 1)}{k!} = \frac{n!}{k!(n - k)!}.$$

Теорема доведена.

Довільна k -елементна підмножина множини A називається **комбінацією** або **сполученням**, а число C_n^k — **числом комбінацій** або **сполучень** із n елементів по k елементів.

Числа C_n^k називають **біноміальними коефіцієнтами**. Зміст цієї назви буде зрозумілим дещо пізніше.

Біноміальні коефіцієнти мають цікаву геометричну інтерпретацію. Нехай маємо прямокутну шахову дошку розміром m на n , розміщену на координатній площині так, як це показано на рис. 1.2.1.

Ця дошка складається з $m \cdot n$ елементарних квадратів, розділених $n - 1$ горизонтальною і $m - 1$ вертикальною лініями. Визначимо, скількома різними найкоротшими шляхами можна попасті з точки $(0, 0)$ в точку (m, n) на цій дошці.

Кожний найкоротший шлях, який веде з точки $(0, 0)$ в точку (m, n) , складається, очевидно, з $m + n$ сторін елементарних квадратів, серед яких m горизонтальних і n вертикальних. Ці шляхи розрізняються лише числом вертикальних і горизонтальних сторін. Значить, загальна кількість шляхів дорівнює числу способів, якими з $m + n$ сторін можна вибрати n вертикальних, тобто це число дорівнює C_{m+n}^n .

Зауважимо, що можна було б вести підрахунки не за вертикальними сторонами, а за горизонтальними. Тобто існує C_{m+n}^m різних найкоротших шляхів і, отже, справедлива рівність

$$C_{m+n}^n = C_{m+n}^m \quad (\text{формула симетрії}).$$

Наслідок 1.2.1.

$$C_n^k = C_{n-1}^k + C_{n-1}^{k-1} \quad (\text{формула додавання}). \quad (1.2.2)$$

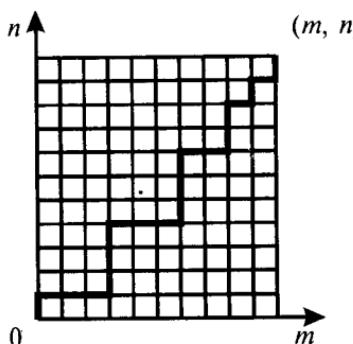


Рис. 1.2.1. Прямоугольна шахова дошка

Д о в е д е н н я.

$$\begin{aligned} C_{n-1}^k + C_{n-1}^{k-1} &= \\ &= \frac{(n-1)!}{k!(n-k-1)!} + \frac{(n-1)!}{(k-1)!(n-k)!} = \\ &= (n-1)! \left(\frac{n-k}{k!(n-k)!} + \frac{k}{k!(n-k)!} \right) = \\ &= \frac{(n-1)! \cdot n}{k!(n-k)!} = \frac{n!}{k!(n-k)!} = C_n^k. \end{aligned}$$

Наслідок доведено.

Приклади 1.2.2

1. Збірна команда університету з волейболу налічує 15 чоловік. Скільки різних варіантів повинен розглянути тренер перед грою, щоб заявити список гравців на гру?

Розв'язок. Число гравців волейбольної команди дорівнює шести. Значить, число всіх можливих варіантів — це число різних підмножин, які складаються з шести елементів у множині з 15-ти елементів. Отже, за теоремою 2 маємо

$$C_{15}^6 = \frac{15!}{6!9!} = \frac{15 \cdot 14 \cdot 13 \cdot 12 \cdot 11 \cdot 10}{6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1} = 5 \cdot 7 \cdot 13 \cdot 11 = 5005.$$

2. У скількох точках перетинаються діагоналі випуклого десятикутника, якщо ніякі три з них не перетинаються в одній точці?

Розв'язок. Кожній точці перетину двох різних діагоналей відповідають чотири вершини десятикутника, а кожним чотирьом вершинам — одна точка перетину. Таким чином, число всіх точок перетину дорівнює числу способів, якими з 10-ти вершин можна вибрати чотири вершини, тобто

$$C_{10}^4 = \frac{10!}{4!6!} = \frac{10 \cdot 9 \cdot 8 \cdot 7}{4 \cdot 3 \cdot 2} = 210$$

точок перетину.

3. ЧИСЛО ПІДМНОЖИН ДАНОЇ МНОЖИНИ

Нехай $A = \{a_1, a_2, \dots, a_n\}$ — деяка скінчена множина, елементи якої перенумеровані. При роботі із скінченними множинами на обчислювальних машинах такі множини часто задають за допомогою **характеристичних векторів**. Нехай $A' \subseteq A$ — довільна підмножина множини A . Характеристичний вектор $v(A') = (v_1, v_2, \dots, v_n)$ для множини A' визначається за допомогою такої відповідності:

$$v_i = \begin{cases} 1, & \text{якщо елемент } a_i \text{ із } A \text{ належить множині } A'; \\ 0, & \text{якщо елемент } a_i \text{ із } A \text{ не належить множині } A'. \end{cases}$$

Наприклад, якщо $A = \{a, b, c, d, e, f\}$ і $A' = \{a, e, f\}$, то $v(A') = (1, 0, 0, 0, 1, 1)$.

Теорема 1.2.2. $A' = A'' \Leftrightarrow v(A') = v(A'')$, де A', A'' — деякі підмножини множини A .

Доведення. Нехай v' і v'' — характеристичні вектори множин A' і A'' відповідно. Якщо $A' = A''$, то для всякого i маємо $v'_i = v''_i$. Звідси випливає, що $v'(A') = v''(A'')$.

Навпаки, якщо $v'(A') = v''(A'')$, то для всякого i маємо $v'_i = v''_i$. За визначенням характеристичного вектора, якщо $v'_i = v''_i = 1$, то a_i належить як підмножині A' , так і підмножині A'' , а якщо $v'_i = v''_i = 0$, то елемент a_i не належить ні тій, ні іншій підмножині. Звідси випливає, що підмножини A' і A'' складаються з одних і тих же елементів, тобто $A' = A''$.

Теорема доведена.

Наслідок 1.2.2. Число різних підмножин n -елементної множини дорівнює 2^n .

Дійсно, оскільки число компонент характеристичного вектора дорівнює n і кожна компонента може приймати одне з двох значень — 0 або 1, то число різних можливих характеристичних векторів за основним правилом комбінаторики дорівнює $2 \cdot 2 \cdot \dots \cdot 2 = 2^n$.

Наслідок 1.2.3.

$$\sum_{k=0}^n C_n^k = 2^n,$$

де $0! = 1$.

Дійсно, оскільки C_n^k — число різних k -елементних підмножин n -елементної множини, то сума всіх таких чисел становить число всіх підмножин даної n -елементної множини.

4. ПЕРЕСТАНОВКИ І РОЗМІЩЕННЯ УПОРЯДКОВАНИХ МНОЖИН

Комбінаторні формули, які були встановлені вище, не залежать від порядку елементів у множинах. Формули, які будуть встановлені нижче, відносяться до упорядкованих множин. Дві упорядковані множини вважаються різними, якщо вони розрізняються між собою або своїми елементами, або порядком елементів. Ясно, що коли множина має більше одного елемента, то її можна упорядкувати більше ніж одним способом. Розглянемо скілько-ма різними способами можна упорядкувати скінченну n -елементну множину A .

Упорядковані множини, які відрізняються одна від одної лише порядком своїх елементів (тобто можуть бути одержані з однієї і тієї ж множини лише перестановкою своїх елементів), називаються *перестановками*. Позначимо число всіх перестановок n -елементної множини P_n .

Теорема 1.2.3. $P_n = n!$

Д о в е д е н н я. Будемо послідовно вибирати елементи множини A і розміщувати їх у заданому порядку на n місцях. На перше місце покладаємо будь-який із n елементів множини A , тобто маємо n можливостей. Після того як перше місце заповнилось, на друге місце можна покласти будь-який із $n - 1$ елемента і т. д. За основним правилом комбінаторики маємо $P_n = n \cdot (n - 1) \cdot \dots \cdot 3 \cdot 2 \cdot 1 = n!$ можливостей упорядкування n -елементної множини A . \blacktriangleleft

Приклади 1.2.3

1. Скількома різними способами можна розмістити п'ять книжок на книжковій полиці?

Р о з в ' я з о к. Шукане число розміщень є числом способів упорядкування множини з п'яти елементів. Значить, це число дорівнює $P_5 = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$.

2. Скількома способами можна упорядкувати n -елементну множину $A = \{1, 2, \dots, n\}$ ($n \geq 2$) так, щоб останні два елементи були $n - 1$ і n ?

Р о з в ' я з о к. Число способів буде $P_{n-2} = (n - 2)!$ \blacktriangleleft

5. ПЕРЕСТАНОВКИ З ПОВТОРЕННЯМИ

Розглянемо таку задачу. Скількома способами можна представити n -елементну множину у вигляді об'єднання її підмножин

A_1, A_2, \dots, A_m , що попарно не перетинаються, так що $|A_1| = k_1$, $|A_2| = k_2$, ..., $|A_m| = k_m$, $k_i \geq 0$, $k_1 + k_2 + \dots + k_m = n$.

Поставлену задачу можна розв'язати таким чином. Візьмемо довільну k_1 -елементну підмножину A_1 множини A (це можна зробити $C_n^{k_1}$ способами); серед $n - k_1$ елементів, що залишилися, візьмемо k_2 -елементну підмножину A_2 множини A (це можна зробити $C_{n-k_1}^{k_2}$ способами) і т. д. За основним правилом комбінаторики маємо, що число шуканих відношень еквівалентності становить

$$C_n^{k_1} \cdot C_{n-k_1}^{k_2} \cdots C_{n-k_1-k_2-\dots-k_{m-1}}^{k_m} = \frac{n!}{k_1!(n-k_1)!} \cdot \frac{(n-k_1)!}{k_2!(n-k_1-k_2)!} \cdots$$

$$\cdot \frac{(n-k_1-k_2-\dots-k_{m-1})!}{k_m!(n-k_1-k_2-\dots-k_{m-1}-k_m)!} = \frac{n!}{k_1!k_2!\dots k_m!}.$$

Таким чином, має місце теорема.

Теорема 1.2.4. Нехай k_1, k_2, \dots, k_m — деякі натуральні числа, такі, що $k_1 + k_2 + \dots + k_m = n$. Кількість способів, якими можна побудувати розбиття n -елементної множини на класи A_1, A_2, \dots, A_m , число елементів яких відповідно k_1, k_2, \dots, k_m , становить

$$C_n(k_1, k_2, \dots, k_m) = \frac{n!}{k_1!k_2!\dots k_m!}. \quad (1.2.3)$$

Числа $C_n(k_1, k_2, \dots, k_m)$ називаються **поліноміальними коефіцієнтами**.

Розглянемо одну корисну інтерпретацію теореми 1.2.4. Нехай маємо деякий алфавіт $X = \{a, b, c, \dots, d\}$ з m символів і множину з n символів цього алфавіту, причому серед n елементів цієї множини є k_1 — букв a , k_2 — букв b , ..., k_m — букв d і $k_1 + k_2 + \dots + k_m = n$. Необхідно визначити кількість різних слів, які можна побудувати з цих n букв.

Перенумеруємо місця, на яких стоять букви, числами від 1 до n . Кожне слово однозначно визначається множинами A_1 (номери місць, на яких стоїть буква a) A_2 (номери місць, на яких стоїть буква b), ..., A_m (номери місць, на яких стоїть буква d). Отже, число різних слів дорівнює числу різних представлень n -елементної множини $\{1, 2, \dots, n\}$ у вигляді об'єднання її підмножин A_1, A_2, \dots, A_m , тобто

$$C_n(k_1, k_2, \dots, k_m) = \frac{n!}{k_1!k_2!\dots k_m!}.$$

З цієї задачі випливає інше формулювання попередньої теореми.

Теорема 1.2.5. Число різних перестановок, які можна побудувати з n елементів, серед яких знаходиться k_1 елементів першого типу, k_2 елементів другого типу, ..., k_m елементів m -го типу, дорівнює $C_n(k_1, k_2, \dots, k_m)$.

Приклади 1.2.4

1. Скільки різних слів можна побудувати перестановкою букв в слові "лаваш"?

Розв'язок. Слово "лаваш" містить по одному екземпляру букв л, в, ш і два екземпляри букви а, а загальна кількість букв 5. За формулою (1.2.3) знаходимо

$$\frac{5!}{2! 1! 1! 1!} = 5 \cdot 4 \cdot 3 = 60.$$

2. Скільки слів довжини 8 можна скласти з букв а і б, таких, що кількість букв а в цих словах не перевищує три?

Розв'язок. Такими словами будуть всі слова, які не мають ні однієї букви а, всі слова, які мають лише одну букву а, всі слова, які мають дві букви а, і, нарешті, всі слова, які мають три букви а, тобто загальна кількість слів становить

$$C_8(0,8) + C_8(1,7) + C_8(2,6) + C_8(3,5) = \frac{8!}{0!8!} + \frac{8!}{1!7!} + \frac{8!}{2!6!} + \frac{8!}{3!5!} = \\ = 1 + 8 + 28 + 56 = 93. \blacksquare$$

6. РОЗМІЩЕННЯ ЕЛЕМЕНТІВ МНОЖИНІ

Нехай дана деяка неупорядкована n -елементна множина A . Скільки різних упорядкованих k -елементних підмножин може мати множина A ? Розглянемо два можливі варіанти цієї задачі:

а) підмножина має k різних між собою елементів;

б) підмножина має k не обов'язково різних між собою елементів.

Отже, в задачі "а" підмножини задаються ненадлишково, а в задачі "б" — надлишково, але число всіх різних елементів підмножини разом з числом всіх екземплярів кожного з її елементів дорівнює k .

Упорядкована k -елементна підмножина множини A , всі елементи якої різні, називається **розміщенням без повторень**, а всяка упорядкована k -елементна підмножина множини A , всі k елементів якої не обов'язково різні, називається **розміщенням з повтореннями**. Зауважимо, що в першому випадку $k \leq n$, причому якщо $k = n$, то в цьому випадку розміщення є перестановкою. У

другому випадку k не обов'язково має бути менше n , тобто можливо, що $k \geq n$.

6.1. Роз'язання першої задачі. Оскільки множина A неупорядкована, то всяка її k -елементна підмножина може бути упорядкована одним із $k!$ способів, а число всіх можливих різних k -елементних підмножин множини A дорівнює C_n^k . Отже, число всіх можливих розміщень із n елементів по k дорівнює $k! C_n^k$, тобто має місце така теорема.

Теорема 1.2.6. Кількість упорядкованих k -елементних підмножин n -елементної множини, всі k елементів якої різні, становить

$$A_n^k = k! C_n^k = \frac{n!}{(n-k)!} = n \cdot (n-1) \cdot \dots \cdot (n-k+1).$$

Приклади 1.2.5

1. Студенту необхідно скласти три екзамени протягом семи днів. Скількома способами це можна зробити?

Розв'язок. Шукане число способів дорівнює кількості трьохелементних упорядкованих підмножин множини з семи елементів, тобто існує $A_7^3 = 7 \cdot 6 \cdot 5 = 210$ способів.

Якщо відомо, що останній екзамен буде складатися на сьомий день, то число способів становить $3 \cdot A_5^2 = 3 \cdot 6 \cdot 5 = 90$.

2. Скількома різними способами можна розмістити п'ять студентів в аудиторії, яка має 20 місць?

Розв'язок. Шукане число способів дорівнює числу розміщень із 20 елементів, по 5 елементів, тобто

$$A_{20}^5 = 20 \cdot 19 \cdot 18 \cdot 17 \cdot 16 = 1860480. \blacksquare$$

6.2. Роз'язання другої задачі. Нехай $B = \{b_1, b_2, b_k\}$ — деяка скінчenna k -елементна множина, а $A = \{a_1, a_2, \dots, a_n\}$ — n -елементна множина і $f: B \rightarrow A$ — функція з B в A . Як відомо, функцію f можна задати за допомогою таблиці значень

b_1	b_2	\dots	b_k
a_{i1}	a_{i2}	\dots	a_{ik}

,

де $a_{ij} = f(b_j)$, $j = 1, 2, \dots, k$. Тепер нашу задачу можна сформулювати так: скільки існує функцій із множини B в множину A ? В такому формульованні задача розв'язується досить просто.

Умовимося називати кортежом довжини k елементи виду (a_1, a_2, \dots, a_k) , де a_i — не обов'язково різні елементи деякої скінчен-

ної множини A . Оскільки кожний елемент a_{i_j} може бути будь-яким елементом множини A , то число різних кортежів виду $(a_{i_1}, a_{i_2}, \dots, a_{i_k})$ може бути n^k .

Теорема 1.2.7. Кількість різних упорядкованих k -елементних підмножин n -елементної множини, всі k елементів якої не обов'язково різні між собою, дорівнює n^k .

Приклади 1.2.6

1. Скільки різних слів можна скласти:
 - а) в алфавіті $X = \{0, 1\}$ з восьми символів;
 - б) в алфавіті $X = \{0, 1\}$ з 16 символів?

Розв'язок: а) всіх таких слів буде стільки, скільки існує відображення восьмиелементної множини в множину з двох елементів, тобто за теоремою 1.2.7 — $2^8 = 256$ слів;

б) у цьому алфавіті маємо $2^{16} = 65536$ слів. \blacktriangleleft

7. КОМБІНАЦІЙ ЕЛЕМЕНТІВ З ПОВТОРЕННЯМИ

Нехай маємо неупорядковану n -елементну множину A , елементи якої розбиті на n класів (у кожному класі є по одному елементу), які будуть називатися **типами** елементів. **Комбінацією** з n елементів по m елементів з повтореннями називається m -елементна підмножина множини A , кожний елемент якої належить одному з n типів. Сукупність таких комбінацій називають комбінаціями з n елементів по m .

Теорема 1.2.8. Кількість різних комбінацій із n елементів по m елементів з повтореннями становить

$$C_{n+m-1}^{n-1} = C_{n+m-1}^m.$$

Доведення. “Закодуємо” кожну комбінацію таким чином: якщо комбінація містить k_1 елементів першого типу, то записуємо підряд k_1 одиниць, ставимо нуль і після нього ставимо підряд k_2 одиниць, якщо комбінація містить k_2 елементів другого типу, ставимо нуль і т. д. Наприклад, якщо $A = \{a, b, c, d\}$, то комбінаціям по два елементи з повтореннями відповідають пари $\{a, a\}, \{a, b\}, \{a, c\}, \{a, d\}, \{b, b\}, \{b, c\}, \{b, d\}, \{c, c\}, \{c, d\}, \{d, d\}$, а їх “кодами” будуть відповідно

11000, 10100, 10010, 10001, ..., 00011.

Неважко переконатися, що між “кодами” і комбінаціями існує взаємно однозначна відповідність (переконайтесь). Таким чином, кожній комбінації з n елементів по m відповідає послі-

довність із m одиниць і $n - 1$ нулів (в розглянутому прикладі $n = 4$, $m = 2$). Отже, число всіх комбінацій із n елементів по m з повтореннями дорівнює числу послідовностей, що складаються із m одиниць і $n - 1$ нуля, тобто $C_{n+m-1}^m = C_{n+m-1}^{n-1}$.

Теорема доведена.

Приклад 1.2.7

1. Скільки цілих невід'ємних розв'язків має рівняння

$$x_1 + x_2 + \dots + x_n = m? \quad (1.2.4)$$

Розв'язок. Розв'язки даного рівняння можна інтерпретувати так. Якщо маємо цілі невід'ємні числа x_1, x_2, \dots, x_n , такі, що $x_1 + x_2 + \dots + x_n = m$, то можна скласти комбінацію з n елементів по m , взявши x_1 елементів першого типу, x_2 елементів другого типу, ..., x_n елементів n -го типу.

Навпаки, маючи комбінацію з n по m елементів, одержимо розв'язок рівняння (1.2.4) (x_1, x_2, x_n — число елементів першого, другого і відповідно n -го типу), де всі x_i невід'ємні ($i = 1, 2, \dots, n$). Таким чином, між множиною всіх невід'ємних розв'язків рівняння (1.2.4) і множиною всіх комбінацій із n елементів по m встановлюється взаємно однозначна відповідність. Отже, число цілих невід'ємних розв'язків рівняння (1.2.4) становить

$$C_{n+m-1}^{n-1} = C_{n+m-1}^m.$$

Наприклад, якщо $x_1 + x_2 + x_3 + x_4 = 10$, то це рівняння має

$$C_{10+4-1}^9 = C_{13}^4 = \frac{13!}{4! 9!} = 715$$

цілих невід'ємних розв'язків. ▲

8. БІНОМ НЬЮТОНА

З елементарної математики добре відомі формули скороченого множення:

$$(a+b)^2 = a^2 + 2ab + b^2 \quad i \quad (a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3.$$

Ці формули можна записати і так:

$$(a+b)^2 = C_2^0 a^2 b^0 + C_2^1 a b + C_2^2 a^0 b^2,$$

$$(a+b)^3 = C_3^0 a^3 b^0 + C_3^1 a^2 b + C_3^2 a b^2 + C_3^3 a^0 b^3.$$

Виявляється, що існує і загальна закономірність.

Теорема 1.2.9. Справедлива рівність

$$(a+b)^n = C_n^0 a^n b^0 + C_n^1 a^{n-1} b + C_n^2 a^{n-2} b^2 + \dots + C_n^n a^0 b^n. \quad (1.2.5)$$

Доведення проводиться індукцією за числом n .

При $n = 1$ $(a+b)^1 = C_1^0 a^1 b^0 + C_1^0 a^1 b^0 + C_1^1 a^0 b^1 = a+b$. Отже, база індукції доведена.

Нехай теорема справедлива для n . Покажемо, що вона справедлива і для $n+1$. За припущенням індукції маємо

$$\begin{aligned} (a+b)^{n+1} &= (a+b)^n(a+b) = (C_n^0 a^n b^0 + C_n^1 a^{n-1} b + \dots + C_n^n a^0 b^n)(a+b) = \\ &= C_n^0 a^{n+1} b^0 + (C_n^1 a^n b + C_n^0 a^n b) + (C_n^2 a^{n-1} b^2 + C_n^1 a^{n-1} b^2) + \dots + C_n^n a^0 b^{n+1} = \\ &= C_{n+1}^0 a^{n+1} b^0 + C_{n+1}^1 a^n b + \dots + C_{n+1}^n a b^n + C_{n+1}^{n+1} a^0 b^{n+1}, \end{aligned}$$

згідно з наслідком 1.2.1 і тим, що $C_n^0 = C_{n+1}^0 = 1$.

Теорема доведена.

Рівність (1.2.5) називають **біномом Ньютона**. З рівності, встановленої в наслідку 1.2.1, випливає, що біноміальні коефіцієнти (тепер ясно, чому ми так назвали їх раніше) можна вписати у вигляді трикутної таблиці, яка носить назву **трикутника Паскаля**:

1	1	1		$n = 1$
1	2	1		$n = 2$
1	3	3	1	$n = 3$
1	4	6	4	1
1	5	10	10	5
				$n = 4$
				$n = 5$

У n -му рядку трикутника Паскаля стоять коефіцієнти розкладу (1.2.5), причому кожний коефіцієнт, крім двох крайніх, які дорівнюють 1, — це сума відповідних коефіцієнтів із попереднього рядка. Цей трикутник має ще деякі корисні властивості, з якими можна детальніше познайомитися в [10, 11, 28, 60, 62, 73].

Наступна теорема є узагальненням бінома Ньютона і дає відповідь на те, як розкривати дужки при обчисленні виразу виду $(a_1 + a_2 + \dots + a_k)^n$.

Поліноміальна теорема.

$$(a_1 + a_2 + \dots + a_k)^n = \sum_{\substack{r_1 \geq 0, \dots, r_k \geq 0 \\ r_1 + \dots + r_k = n}} C_n(r_1, r_2, \dots, r_k) a_1^{r_1} a_2^{r_2} \dots a_k^{r_k}.$$

Доведення. Перемножимо послідовно $a_1 + a_2 + \dots + a_k$ n разів. В результаті одержимо k^n доданків $d_1 d_2 \dots d_n$, де кожний множник d_i дорівнює або a_1 , або a_2 , ..., або a_k . Нехай $B(r_1, r_2, \dots, r_k)$ означає множину всіх таких доданків, в яких a_1 зустрічається r_1 разів, $a_2 - r_2$ разів, ..., $a_k - r_k$ разів. Як ми знаємо

(див. теорему 1.2.4), число таких доданків дорівнює числу способів розбиття множини $\{1, 2, \dots, n\}$ на k підмножин B_1, B_2, \dots, B_k так, щоб в множині B_s було r_s елементів ($r_s \geq 0$), тобто дорівнює $C_n(r_1, r_2, \dots, r_k)$, де $r_1 + r_2 + \dots + r_k = n$, а B_s — множина тих значень i , для яких $d_i = a_s$. Отже,

$$(a_1 + a_2 + \dots + a_k)^n = \sum_{\substack{r_1 \geq 0, \dots, r_k \geq 0 \\ r_1 + \dots + r_k = n}} C_n(r_1, r_2, \dots, r_k) a_1^{r_1} a_2^{r_2} \dots a_k^{r_k}.$$

Теорема доведена.

$$\text{Наслідок 1.2.4. } (a + b)^n = \sum_{k=0}^n C_n^k a^{n-k} b^k.$$

Дійсно, якщо покласти $r_1 = k$, то $r_2 = n - k$ і з поліноміальної теореми випливає даний наслідок.

9. ВЛАСТИВОСТІ БІНОМІАЛЬНИХ КОЕФІЦІЄНТІВ

Раніше ми вже встановили деякі властивості біноміальних коефіцієнтів. Нагадаємо їх:

(а) формула симетрії

$$C_{n+m}^n = C_{n+m}^m;$$

(б) формула додавання

$$C_n^k = C_{n-1}^k + C_{n-1}^{k-1};$$

(в) формула суми всіх біноміальних коефіцієнтів

$$C_0^0 - C_n^1 + C_n^2 - \dots + C_n^n = 2^n.$$

Користуючись цими властивостями, неважко одержати іще ряд формул. Наприклад, якщо покласти в біномі Ньютона $a = 1$, $b = -1$, то одержимо таку властивість:

$$(г) \quad C_0^0 - C_n^1 + C_n^2 - \dots + (-1)^n C_n^n = 0.$$

Наступну формулу легко можна встановити, виконуючи обчислення

(д) формула винесення за дужки

$$C_n^k = \frac{n}{k} \cdot C_{n-1}^{k-1}.$$

Наведемо ще деякі корисні властивості біноміальних коефіцієнтів:

- (e) $C_n^0 + C_{n+1}^1 + \dots + C_{n+k}^n = C_{n+k+1}^n$;
 (e) $C_{n-1}^{r-1} + C_{n-2}^{r-1} + \dots + C_{r-1}^{r-1} = C_n^r$;
 (ж) $C_n^0 + 2C_n^1 + \dots + nC_n^n = n2^{n-1}$;
 (з) $C_n^0 C_m^k + C_n^1 C_m^{k-1} + \dots + C_n^k C_m^0 = C_{n+m}^k$;
 (и) $C_n^m C_k^0 + C_{n-1}^{m-1} C_{k+1}^1 + \dots + C_{n-m}^0 C_{k+m}^m = C_{n+k+1}^m$;
 (і) $(C_n^0)^2 + (C_n^1)^2 + \dots + (C_n^n)^2 = C_{2n}^n$.

На закінчення даного розділу зазначимо, що переліченими властивостями біноміальних коефіцієнтів далеко не вичерпуються всі їх властивості. Існує велика кількість властивостей цих коефіцієнтів і виявляються все нові і нові. Як зазначає Кнут у своїй книзі “Искусство программирования для ЭВМ”, т.1, с. 85, за останні роки таких властивостей виявлено стільки, що вони задовольняють лише тих людей, які їх встановили. Тому дати вичерпну відповідь про властивості біноміальних коефіцієнтів немає ніякої можливості.

Переходимо до розгляду основних методів комбінаторного аналізу.

10. МЕТОД РЕКУРЕНТНИХ СПІВВІДНОШЕНЬ

Метод рекурентних співвідношень, як уже відомо, дозволяє знаходити значення деякої функції для заданої величини аргументу через менші значення аргументів. Щодо комбінаторики цей метод дає можливість знаходити розв'язок комбінаторної задачі для n предметів через розв'язок аналогічної задачі з меншим числом предметів за допомогою деякого співвідношення, яке називається *рекурентним співвідношенням*. Метод рекурентних співвідношень добре відомий з курсу шкільної математики, де він застосовувався при визначенні сум арифметичної і геометричної прогресій та при визначенні n -го члену цих прогресій.

Розглянемо ще деякі приклади.

1. Ряд чисел Фібоначчі. Цей ряд задається такими співвідношеннями:

$$a_1 = a(1) = 1, a_2 = a(2) = 1, a_n = a_{n-1} + a_{n-2}, n > 2.$$

Користуючись ними, одержимо послідовність чисел, яка відома під назвою *послідовності чисел Фібоначчі*.

2. Квадрати натуральних чисел. Розглянемо послідовність квадратів натуральних чисел $a_1 = a(1) = 1^2, a_2 = a(2) = 2^2, \dots, a_n = a(n) = n^2, \dots$ Користуючись формuloю скороченого множення

$(a + b)^2 = a^2 + 2ab + b^2$, одержуємо таку рекурентну формулу:
 $a_{n+1} = (n + 1)^2 = n^2 + 2n + 1 = a_n + 2n + 1$.

Використовуючи її, можна знаходити квадрати чисел без складних обчислень. Наприклад, коли відомо, що $15^2 = 225$, то легко знайти 16^2 . Дійсно, $16^2 = 225 + 2 \cdot 15 + 1 = 225 + 31 = 256$.

Аналогічно можна одержати формулу і для обчислення кубів натуральних чисел.

3. Комбінації з повторенням. Нехай число комбінацій із n елементів по k елементів з повтореннями дорівнює f_n^k . Кожна комбінація з n по k елементів або включає, або не включає елемент a_n . Число комбінацій, які не включають елемент a_n , дорівнює f_{n-1}^k . Кожна комбінація може бути одержана приєднанням до елемента a_n деякої комбінації з n елементів по $k - 1$ елементів, тобто існує f_{n-1}^{k-1} комбінацій. Отже, всіх комбінацій буде

$$f_n^k = f_{n-1}^{k-1} + f_{n-1}^k.$$

Побудоване рекурентне співвідношення дає можливість знайти f_n^k . Дійсно,

$$f_n^k = f_n^{k-1} + (f_{n-1}^{k-1} + f_{n-2}^k) = f_n^{k-1} + f_{n-1}^{k-1} + \dots + f_2^{k-1} + f_1^k. \quad (1.2.6)$$

Очевидно, що $f_n^1 = n$ і $f_1^k = 1$.

Поклавши в формулі (1.2.6) $k = 2$, одержимо

$$\begin{aligned} f_n^2 &= f_n^1 + f_{n-1}^2 = f_n^1 + f_{n-1}^1 + \dots + f_2^1 + f_1^2 = \\ &= n + (n - 1) + \dots + 2 + 1 = \frac{n(n + 1)}{2} = C_{n+1}^2. \end{aligned}$$

Скориставшись формулою (ε) п. 9 і поклавши в рівнянні (1.2.6) $k = 3$, дістанемо

$$f_n^3 = C_{n+1}^2 + C_n^2 + \dots + C_3^2 + C_2^2 = C_{n+2}^3.$$

Повторюючи ці обчислення k разів, одержуємо остаточну формулу $f_n^k = C_{n+k-1}^k$.

4. Наближені обчислення значень елементарних функцій. Формули Маклорена, які добре відомі з математичного аналізу і які дають розклад елементарних функцій у степеневі ряди в околі деякого значення аргументу з їх області визначення, — важливий приклад використання рекурентних співвідношень. Важливість їх полягає в тому, що вони дають можливість обчислювати значення функції практично з будь-якою наперед заданою точністю. Розглянемо приклади:

a) експонента

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots + \frac{x^n}{n!} + \dots$$

задається такими рекурентними співвідношеннями:

$$s_1 = 1, \quad s_2 = s_1 + \frac{x}{1!}; \quad s_{n+1} = s_n + \frac{x^n}{n!}, \text{ де } n \geq 2;$$

б) синус

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots + (-1)^{\frac{n-1}{2}} \frac{x^n}{n!} + \dots$$

задається такими рекурентними співвідношеннями:

$$s_2 = x, \quad s_4 = s_2 - \frac{x^3}{3!}; \quad s_{2k} = s_{2k-2} + (-1)^{k-1} \frac{x^{2k-1}}{(2k-1)!};$$

в) косинус

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots + (-1)^k \frac{x^{n2k}}{(2k)!} + \dots$$

задається такими рекурентними співвідношеннями:

$$s_1 = 1, \quad s_3 = s_1 - \frac{x^2}{2!}; \quad s_{2k+1} = s_{2k-1} + (-1)^k \frac{x^{2k}}{(2k)!}.$$

Користуючись даними рекурентними співвідношеннями, можна обчислювати значення функцій всіх тригонометричних функцій.

Наведені рекурентні співвідношення відіграють важливу роль при обчисленнях значень даних функцій на ЕОМ, оскільки дають зручний спосіб програмування методів обчислення значень елементарних функцій. Детальніше з методами програмування елементарних функцій можна познайомитися в [12].

11. МЕТОД ВКЛЮЧЕНЬ І ВИЛУЧЕНЬ

Нехай A і B — скінченні множини. З'ясуємо, чому дорівнює число $|A \cup B|$, якщо відомі числа $|A|$ і $|B|$. Основна формула, якою можна скористатися для визначення числа $|A \cup B|$, випливає безпосередньо з означення операції об'єднання множин:

$$|A \cup B| = |A| + |B| - |A \cap B|. \quad (1.2.7)$$

Дійсно, $|A| + |B|$ є числом, яке одержується при підрахунку спочатку всіх елементів множини A , а потім всіх елементів множини B . Але при цьому спільні елементи множин A і B (а їх буде $|A \cap B|$)

$\cap B|$ рахуються двічі, тобто $|A| + |B| - |A \cup B| + |A \cap B|$, звідки і випливає рівність (1.2.7).

Користуючись формулою (1.2.7) і законами М1)–М2) для множин, можна одержати формулу для числа елементів будь-якої скінченної сукупності скінченних множин. Наприклад,

$$\begin{aligned} |A \cup B \cup C| &= |A \cup (B \cup C)| = |A| + |B \cup C| - |A \cap (B \cup C)| = \\ &= |A| + |B| + |C| - |B \cap C| - |(A \cap B) \cup (A \cap C)| = |A| + |B| + |C| - \\ &\quad - |B \cap C| - (|A \cap B| + |A \cap C| - |(A \cap B) \cap (A \cap C)|) = \\ &= |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|. \end{aligned}$$

Справедлива загальна теорема.

Теорема 1.2.10. Якщо A_1, A_2, \dots, A_n — деякі скінченні множини, то

$$\begin{aligned} |A_1 \cup A_2 \cup \dots \cup A_n| &= \sum_{1 \leq i \leq n} |A_i| - \sum_{1 \leq i < j \leq n} |A_i \cap A_j| + \\ &\quad \sum_{1 \leq i < j < k \leq n} |A_i \cap A_j \cap A_k| + \dots + \\ &\quad + (-1)^{n-1} \sum_{1 \leq i < j < k < \dots < l \leq n} |A_i \cap A_j \cap \dots \cap A_l|. \end{aligned} \quad (1.2.8)$$

Д о в е д е н н я. Щоб довести теорему, необхідно показати, що кожний елемент із множини $A_1 \cup A_2 \cup \dots \cup A_n$ враховується в правій частині рівності (1.2.8) лише один раз. Нехай $a \in A_1 \cup \dots \cup A_n$, такий елемент, який входить до складу m множин A_i ($i = 1, 2, \dots, n$). Тоді елемент a підраховується в правій частині (1.2.8) $C_m^1 - C_m^2 + C_m^3 - \dots + (-1)^{m-1} C_m^m$ разів. Але

$$\begin{aligned} C_m^1 - C_m^2 + C_m^3 - \dots + (-1)^{m-1} C_m^m &= 1 - (1 - C_m^1 + C_m^2 - C_m^3 + \\ &\quad + \dots + (-1)^m C_m^m) = 1 - (1 - 1)^m = 1, \end{aligned}$$

а це означає, що кожний елемент a із $A_1 \cup A_2 \cup \dots \cup A_n$ враховується в правій частині рівності (1.2.8) лише один раз.

Теорема доведена.

Метод підрахунку за формулою (1.2.8), який полягає в послідовному виконанні операцій додавання і віднімання, що чергуються між собою, називається **методом включення і вилучення**.

П р и к л а д и 1.2.8

1. У групі студентів кожний студент або блондин, або брюнет. Серед студентів 10 блондинів, решта — брюнети. 12 студентів люблять читати детективи. Серед 12-ти студентів, які люблять читати детективи, 5 блондинів і 7 брюнетів. Скільки чоловік налічує вся група, якщо два брюнети не люблять читати детективи?

Р о з в ' я з о к. Нехай A означає множину блондинів: $|A| = 10$; B — множину брюнетів: $|B| = 9$; C — множину любителів читати детективи: $|C| = 12$. Тоді $A \cap B = \emptyset$, оскільки множини блондинів і брюнетів не можуть мати спільних елементів; $A \cap C = \emptyset$ — множина блондинів, які люблять читати детективи: $|A \cap C| = 5$; $B \cap C$ — множина брюнетів, які люблять читати детективи: $|B \cap C| = 7$. Множина $A \cap B \cap C = \emptyset$. Отже,

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C| = 10 + 9 + 12 - 0 - 5 - 7 + 0 = 19.$$

2. Розглянемо всі перестановки множини чисел $\{1, 2, \dots, n\}$. Необхідно підрахувати кількість всіх перестановок, в яких хоча б одне число стоїть на своєму місці.

Р о з в ' я з о к. Нехай A_i означає сукупність тих перестановок, в яких число i стоїть на i -му місці. Тоді $S = |A_1 \cup A_2 \cup \dots \cup A_n|$ є шуканим числом. Множина $A_1 \cap A_2 \cap \dots \cap A_k$ містить ті перестановки, в яких на місцях i, j, \dots, k стоять відповідно числа i, j, \dots, k , а на інших $n - k$ місцях — решта $n - k$ чисел, які упорядковані. Отже, $|A_1 \cap A_2 \cap \dots \cap A_k| = (n - k)!$, а

$$\sum_{1 \leq i < j < \dots < k \leq n} |A_i \cap A_j \cap \dots \cap A_k| = C_n^k (n - k)! = \frac{n!}{k!}.$$

З рівності (1.2.6) випливає, що

$$S = |A_1 \cap A_2 \cap \dots \cap A_n| = n! \left(\frac{1}{1!} - \frac{1}{2!} + \frac{1}{3!} - \dots + (-1)^{n-1} \frac{1}{n!} \right).$$

3. Нехай a_1, a_2, \dots, a_n — взаємно прості натуральні числа, а p — теж деяке натуральне число. Довести, що кількість натуральних чисел, які не перевищують p і не діляться ні на одне з чисел a_1, a_2, \dots, a_n , дорівнює $p - |A_1 \cup A_2 \cup \dots \cup A_n|$, де $A_i = \{q \in \mathbb{N} \mid q < p \text{ і } q \text{ ділиться на } a_i\}$.

Р о з в ' я з о к. Очевидно, що $|A_i| = \left[\frac{p}{a_i} \right]$, де $[x]$ — найбільше ціле число, яке не перевищує x . Множина $A_i \cap A_j \cap \dots \cap A_k$ — це множина таких чисел $q < p$, які діляться на a_i, a_j, \dots, a_k . Оскільки числа a_i, a_j, \dots, a_k взаємно прості, то

$$|A_i \cap A_j \cap \dots \cap A_k| = \left[\frac{p}{a_i a_j \dots a_k} \right].$$

З рівності (1.2.8) маємо, що кількість всіх чисел, які не перевищують p і діляться хоча б на одне з чисел a_1, a_2, \dots, a_p , становить

$$|A_1 \cup A_2 \cup \dots \cup A_n| = \\ = \sum_{1 \leq i \leq n} \left[\frac{p}{a_i} \right] - \sum_{1 \leq i < j \leq n} \left[\frac{p}{a_i a_j} \right] + \sum_{1 \leq i < j < k \leq n} \left[\frac{p}{a_i a_j a_k} \right] + \dots + (-1)^{n-1} \left[\frac{p}{a_1 a_2 \dots a_n} \right].$$

Отже, кількість чисел, які не перевищують p і не діляться ні на одне з чисел a_1, a_2, \dots, a_n , становить

$$p - |A_1 \cup A_2 \cup \dots \cup A_n| = p - \\ - \sum_{1 \leq i \leq n} \left[\frac{p}{a_i} \right] + \sum_{1 \leq i < j \leq n} \left[\frac{p}{a_i a_j} \right] - \sum_{1 \leq i < j < k \leq n} \left[\frac{p}{a_i a_j a_k} \right] + \dots + (-1)^{n-1} \left[\frac{p}{a_1 a_2 \dots a_n} \right],$$

що й потрібно було встановити. \blacktriangleleft

Теорема 1.2.11. Нехай $|A_1 \cap A_2 \cap \dots \cap A_n|^m$ — число елементів, які входять до складу точно m множин із A_1, A_2, \dots, A_n . Тоді

$$|A_1 \cap A_2 \cap \dots \cap A_n|^m = C_m^m \sum_{1 \leq i_1 < i_2 < \dots < i_m \leq n} |A_{i_1} \cap A_{i_2} \cap \dots \cap A_{i_m}| - \\ - C_{m+1}^m \sum_{1 \leq i_1 < i_2 < \dots < i_{m+1} \leq n} |A_{i_1} \cap A_{i_2} \cap \dots \cap A_{i_{m+1}}| + \dots + \\ + (-1)^{n-m} C_n^m \sum_{1 \leq i_1 < i_2 < i_3 < \dots < i_n \leq n} |A_{i_1} \cap A_{i_2} \cap \dots \cap A_{i_n}|. \quad (1.2.9)$$

Д о в е д е н и я. Нехай a — довільний елемент, який входить до складу k множин із заданої сукупності множин. Для доведення теореми, як і в попередньому випадку, досить показати, що елемент a враховується в правій частині рівності (1.2.9) лише один раз, якщо $k = m$, і жодного разу, якщо $k \neq m$.

Насамперед зазначимо, що коли $k < m$, то елемент a не враховується в (1.2.9) ні разу, а якщо $k = m$, то враховується лише один раз, оскільки a входить лише в одну з множин вигляду $|A_{i_1} \cap A_{i_2} \cap \dots \cap A_{i_k}|$.

Розглянемо випадок, коли $k > m$. При підрахунках за формулою (1.2.9) елемент a враховується C_k^m разів у першій сумі

$$\sum_{1 \leq i_1 < i_2 < i_3 < \dots < i_m \leq k} |A_{i_1} \cap A_{i_2} \cap \dots \cap A_{i_m}|,$$

C_k^{m+1} разів у другій сумі

$$\sum_{1 \leq i_1 < i_2 < i_3 < \dots < i_{m+1} \leq k} |A_{i_1} \cap A_{i_2} \cap \dots \cap A_{i_{m+1}}|$$

і т. д. C_k^k разів у сумі

$$\sum_{1 \leq i_1 < i_2 < i_3 < \dots < i_k \leq k} |A_{i_1} \cap A_{i_2} \cap \dots \cap A_{i_k}|.$$

Решта сум не враховують елемент a , оскільки він входить лише до k множин. Отже, елемент a враховується в (1.2.9):

$$C_m^m C_k^m = C_{m+1}^m C_k^{m+1} + C_{m+2}^m C_k^{m+2} + \dots + (-1)^{k-m} C_k^m C_k^k$$

разів. Оскільки $C_r^m C_n^r = C_n^m C_{n-m}^{n-r}$,

знаходимо

$$\begin{aligned} C_m^m C_k^m &= C_{m+1}^m C_k^{m+1} + C_{m+2}^m C_k^{m+2} + \dots + (-1)^{k-m} C_k^m C_k^k = \\ &= C_k^m (C_{k-m}^{k-m} - C_{k-m}^{k-m-1} + \dots + (-1)^{k-m} C_{k-m}^0) = C_k^m (1-1)^{k-m} = 0. \end{aligned}$$

Теорема доведена.

12. МЕТОД ПРОДУКТИВНИХ ФУНКЦІЙ

Метод продуктивних функцій — це один з найефективніших методів комбінаторного аналізу, оскільки він узагальнює інші методи комбінаторного аналізу. Введемо поняття продуктивної функції.

Нехай маємо деяку послідовність $a_1, a_2, \dots, a_n, \dots$. Формальна сума

$$A(s) = a_0 + a_1 s + a_2 s^2 + \dots + a_n s^n + \dots = \sum_{n=0}^{\infty} a_n s^n \quad (1.2.10)$$

називається продуктивною функцією послідовності $\{a_n\}$.

Ідея методу продуктивних функцій полягає в тому, що для того, щоб знайти всі члени деякої послідовності $\{a_n\}$, користуються такою процедурою: за допомогою рекурентних співвідношень для членів послідовності $\{a_n\}$ обчислюють продуктивну функцію $A(s)$, а потім, розкладаючи її в ряд, знаходять коефіцієнти при s^n , які є a_n .

Як відомо з математичного аналізу, ряди вигляду (1.2.10) можна додавати, віднімати і множити, виконуючи відповідні дії над коефіцієнтами при однакових степенях s . Значить, можна говорити і про суму, різницю і добуток продуктивних функцій.

Приклад 1.2.9

1. Нехай $a_n = c^n$, де c — деяке дійсне число. Тоді

$$A(s) = \sum_{n=0}^{\infty} c^n s^n = \frac{1}{1 - cs}$$

(це буде сума нескінченно спадної геометричної прогресії, якщо $|cs| < 1$).

2. Нехай $a_n = C_m^n$ ($n = 0, 1, 2, \dots, m$). Тоді

$$A(s) = \sum_{n=0}^m C_m^n s^n = (1+s)^m.$$

3. Нехай $a_n = f_m^n$ — число сполучень із m елементів по n . Раніше було доведено, що

$$f_m^n = f_m^{n-1} + f_{m-1}^n, \quad (1.2.11)$$

причому $f_m^0 = 1$. Тоді

$$A_n(s) = \sum_{n=0}^{\infty} f_m^n s^n.$$

Помножимо (1.2.11) на s^n і додамо почленно всі одержані рівності. Маємо

$$\sum_{n=1}^{\infty} f_m^n s^n = s \sum_{n=1}^{\infty} f_m^{n-1} s^{n-1} + \sum_{n=1}^{\infty} f_{m-1}^n s^n = s(A_n(s) - 1) + A_{n-1}(s) - 1,$$

тобто $A_n(s) - 1 = s(A_{n-1}(s) - 1) + A_{n-1}(s) - 1$, або

$$A_n(s) - 1 = \frac{1}{1-s} A_{n-1}(s).$$

Звідси знаходимо, що

$$A_n(s) = \frac{1}{(1-s)^2} A_{n-2}(s) = \dots = \frac{A_1(s)}{(1-s)^{n-1}}.$$

Тепер знайдемо

$$A_1(s) = \sum_{n=0}^{\infty} f_1^n s^n = \sum_{n=0}^{\infty} s^n = \frac{1}{1-s}.$$

Отже, остаточно маємо

$$A_n(s) = \frac{1}{(1-s)^n}.$$

Розкладаючи в ряд одержану функцію за допомогою формули бінома Ньютона (як відомо з математичного аналізу, формула бінома Ньютона справедлива як для дробових, так і для від'ємних показників степеня), знаходимо, що $f_m^n = C_{m-1+n}^n$.⁴

Згортоюю двох послідовностей $\{a_n\}$ і $\{b_n\}$ називається послідовність $\{c_n\}$, загальний член якої має вигляд

$$c_n = a_0 b_n + a_1 b_{n-1} + \dots + a_k b_{n-k} + \dots + a_n b_0 . \quad (1.2.12)$$

Справедливе таке просте твердження.

Теорема 1.2.12. Продуктивна функція згортки двох послідовностей дорівнює добутку продуктивних функцій цих послідовностей.

Д о в е д е н н я. Нехай $\{c_n\}$ — згортка послідовностей $\{a_n\}$ і $\{b_n\}$, а

$$A(s) = \sum_{n=0}^{\infty} a_n s^n , \quad B(s) = \sum_{n=0}^{\infty} b_n s^n , \quad C(s) = \sum_{n=0}^{\infty} c_n s^n$$

є продуктивні функціїй послідовностей $\{a_n\}$, $\{b_n\}$, $\{c_n\}$ відповідно. Необхідно довести, що $C(s) = A(s)B(s)$.

Перемножуючи два ряди: $A(s)$ і $B(s)$, знаходимо, що коефіцієнт при s^n в одержаному добутку дорівнює $a_0 b_n + a_1 b_{n-1} + \dots + a_k b_{n-k} + \dots + a_n b_0$, тобто дорівнює c_n . Отже, $A(s)B(s) = C(s)$.

Теорема доведена.

П р и к л а д 1.2.10

Нехай $a_n = a_{n-1} + a_{n-2}a_1 + \dots + a_1 a_{n-2} + a_{n-1}$ і $A(s) = \sum_{n=0}^{\infty} a_n s^n$ —

продуктивна функція для послідовності $\{a_n\}$, про яку відомо, що $A(s) = 1$. Покладемо $a_0 = 1$. Тоді можемо записати

$$\begin{aligned} a_n &= a_0 a_{n-1} + a_{n-2} a_1 + \dots + a_1 a_{n-2} + a_{n-1} a_0 , \quad n \geq 1 , \\ \sum_{n=1}^{\infty} a_n s^n &= \sum_{n=1}^{\infty} (a_0 a_{n-1} + a_{n-2} a_1 + \dots + a_1 a_{n-2} + a_{n-1} a_0) s^n = \\ &= s \left(\sum_{n=1}^{\infty} (a_0 a_{n-1} + a_{n-2} a_1 + \dots + a_1 a_{n-2} + a_{n-1} a_0) \right) s^{n-1} = s(A(s))^2 . \end{aligned}$$

Оскільки $\sum_{n=1}^{\infty} a_n s^n = A(s) - 1$, маємо таке рівняння: $A(s) - 1 = s(A(s))^2$. Розв'язуючи це рівняння, знаходимо його корені: $A(s) = \frac{1 \pm \sqrt{1 - 4s}}{2s}$.

При $s \rightarrow 0$

$$A(s) = \lim_{s \rightarrow 0} \frac{1 - \sqrt{1 - 4s}}{2s} = 1 , \text{ а } \lim_{s \rightarrow 0} \frac{1 + \sqrt{1 - 4s}}{2s} = \infty .$$

Отже,

$$A(s) = \frac{1 - \sqrt{1 - 4s}}{2s} . \quad (1.2.13)$$

Знайдемо тепер коефіцієнт при s^n в розкладі функції (1.2.13). Скористаємось для цього формулою бінома Ньютона для функції

$$(1 - 4s)^{\frac{1}{2}} = 1 + \sum_{k=1}^{\infty} \frac{\frac{1}{2}\left(\frac{1}{2}-1\right)\cdots\left(\frac{1}{2}-k+1\right)}{k!} (-1)^{k_4 k_3 k} = 1 - \sum_{k=1}^{\infty} C_{2k}^k \frac{1}{2k-1} s^k.$$

Звідси одержуємо

$$A(s) = \frac{1 - \sqrt{1 - 4s}}{2s} = \frac{1}{2} \sum_{k=1}^{\infty} C_{2k}^k \frac{1}{2k-1} s^{k-1} = \frac{1}{2} \sum_{n=0}^{\infty} C_{2n+2}^{n+1} \frac{1}{2n+1} s^n,$$

де $n = k - 1$. Отже,

$$a_n = \frac{1}{2} C_{2n+2}^{n+1} \frac{1}{2n+1} = \frac{1}{(n+1)} C_{2n}^n. \blacktriangleleft$$

Контрольні питання

1. Яка різниця між декартовим квадратом деякої непустої множини A і множиною всіх двохелементних підмножин множини A ?
2. Скільки відношень еквівалентності можна побудувати на множині, яка складається з двох, трьох, чотирьох елементів?
3. Скільки бінарних відношень можна задати на множині з n елементів?
3. Що називається:
 - а) перестановою n -елементної множини?
 - б) сполученням з n елементів по m елементів?
4. Скількома способами можна розмістити три книжки на книжковій полиці?
5. Скільки існує функцій із множини A в множину B , якщо $|A| = m$, а $|B| = n$?

Задачі і вправи

1. Група студентів налічує 25 чоловік. Із них 15 люблять математику, 10 — фізику, 8 — не люблять ні математики, ні фізики. Скільки студентів люблять і математику, і фізику?
2. На зборах студентів-відмінників були як студенти другого, так і студенти третього курсу. Всі вони або любителі прози, або любителі поезії. Студентів-хлопців було 16, а любителів прози — 24. Студентів-дівчат буде рівно стільки, скільки хлопців любителів прози. Скільки студентів було на зборах?

3. У групі з 100 студентів англійською мовою володіють 28 чоловік, німецькою — 30, французькою — 42, англійською і німецькою — 8, англійською і французькою — 10, німецькою і французькою — 5, а всіма трьома мовами володіють 3 студенти. Скільки студентів не знають жодної з названих мов?

4. На кафедрі математики працює сім викладачів. Скількома способами можна скласти комісію з трьох чоловік для прийому “хвостів”?

5. В шаховому турнірі брали участь 30 чоловік, і кожні два шахісти зіграли між собою лише один раз. Скільки партій було зіграно в турнірі?

6. Скільки існує п'ятизначних чисел, в яких кожна наступна цифра:

- а) менша за попередню;
- б) більша за попередню.

7. На площині проведено 10 прямих ліній так, що ніякі дві з них не паралельні між собою і ніякі три з них не перетинаються в одній точці. Знайти:

- а) число точок перетину цих прямих;
- б) число трикутників, які утворюють ці прямі;
- в) на скільки частин ділять площину ці прямі.

8. Скільки прямих можна провести через n точок, якщо ніякі три з них не лежать на одній прямій?

9. У корзині знаходиться p білих і q чорних м'ячів. Скількома способами можна викласти ці м'ячі в ряд так, щоб ніякі два чорних м'ячі не були поряд?

10. Скількома способами можна:

а) упорядкувати множину $\{1, 2, \dots, 2n\}$ так, щоб кожне парне число мало парний номер?

б) розмістити 8 тур на шахівниці так, щоб вони не били одна одні?

в) розмістити 100 книжок на книжковій полиці (звідси буде видно, наскільки необхідним засобом є каталоги в бібліотеках)?

11. В змаганнях по метанню списа беруть участь чотири спортсмени (A, B, C, D). Скількома способами іх можна розмістити в списку виходів у сектор для метання, якщо спортсмен B не може виходити раніше спортсмена A ?

12. Скільки слів із п'яти букв можна скласти, якщо $X = \{a, b, c, d\}$ і буква a зустрічається в слові не більше двох разів, буква b — не більше одного разу і буква c — не більше трьох разів?

13. Нехай $X = \{a, b, c, d\}$ — алфавіт. Слово $p = x_1x_2\dots x_n$ в алфавіті X називається **паліндромом**, якщо слово $p' = x_nx_{n-1}\dots x_1$ дорівнює p . Скільки паліндромів в алфавіті X існує серед слів із п'яти букв?

14. Скільки різних слів можна скласти перестановкою букв у слові “чачача”?

15. Скільки цілих додатних розв'язків має рівняння $x_1 + x_2 + \dots + x_m = n$?

16. Обчислити:

- а) $(a + b + c)^2$;
- б) $(a + b + c)^3$?

17. Знайти коефіцієнт при:

- а) x^5 в розкладі $(1 + x)^7$;

б) x^{17} в розкладі $(1 + x^5)^7$.

18. Показати, що сума $C_p^1 + C_p^2 + \dots + C_p^{p-1}$ ділиться на p , де p — просте число.

19. Довести, що сума всіх поліноміальних коефіцієнтів дорівнює k^n .

20. Довести малу теорему Ферма, тобто, що $a^p - a$ ділиться на p , де a — довільне ціле число, p — просте число.

21. Знайти суми:

а) $C_n^0 + C_n^1 + \dots + (-1)^m C_n^m$;

б) $C_n^0 + C_n^2 + C_n^4 + \dots$;

в) $C_n^0 + C_n^2 + C_n^4 + \dots$.

22. Довести властивості біноміальних коефіцієнтів “д” — “і”.

P o z d i l 2

ОСНОВНІ ПОНЯТТЯ ЗАГАЛЬНОЇ АЛГЕБРИ

У цьому розділі розглядаються основні алгебраїчні поняття, які знаходять широке застосування в різних галузях науки і техніки. До таких понять належать насамперед поняття універсальної алгебри, гомоморфізму та ізоморфізму універсальних алгебр, вільної алгебри (вільні напівгрупи, групи, кільця, структури, булеві алгебри тощо). В кінці розділу вводиться поняття багатоосновної алгебри та поняття підалгебри, гомоморфізму та ізоморфізму багатоосновних алгебр.

§ 2.1. УНІВЕРСАЛЬНІ АЛГЕБРИ

1. ЗАГАЛЬНІ ВІДОМОСТІ

Універсальною Ω -алгеброю (або просто алгеброю) називається система $G(A, \Omega)$, яка складається з деякої непустої множини A (**основна множина алгебри**, або **носій алгебри**) і множини визначених на A операцій $\Omega = \{ \omega_1^{k_1}, \omega_2^{k_2}, \dots, \omega_n^{k_n}, \dots \}$ (**сигнатура алгебри**), де k_i — арність операції ω_i , $k_i \in \mathbb{N}$, $\text{ар}(\omega_i) = k_i$, $i = 1, \dots, n, \dots$. Операції з множини Ω називаються **основними операціями алгебри**.

Нехай $G = (A, \Omega)$ — довільна алгебра, $\omega \in \Omega$, $\text{ар}(\omega) = n$ і $A' \subseteq A$. Підмножина A' називається **замкнутою** відносно операції ω , якщо для довільних a_1, \dots, a_n із A' істинне $\omega(a_1, \dots, a_n) \in A'$. Система (A', Ω) називається **підалгеброю** алгебри (A, Ω) , якщо $A' \subseteq A$ і A' замкнута відносно будь-якої основної операції алгебри $G = (A, \Omega)$.

З означення підалгебри випливає таке просте твердження.

Теорема 2.1.1. *Перетин довільної сукупності підалгебр універсальної алгебри $G = (A, \Omega)$, коли він не пустий, буде підалгеброю цієї алгебри.*

Д о в е д е н н я. Нехай $\omega \in \Omega$, $\text{ar}(\omega) = n$ і $a_1, a_2, \dots, a_n \in G_1 \cap G_2 \cap \dots \cap G_n \cap \dots$ Тоді внаслідок замкнутості носіїв підалгебр відносно операцій із Ω маємо $\omega(a_1, \dots, a_n) \in G_i$, $i = 1, 2, \dots, n, \dots$ Отже, $\omega(a_1, \dots, a_n) \in G_1 \cap G_2 \cap \dots \cap G_n \cap \dots$, тобто перетин теж замкнутий відносно операції ω із Ω .

Наслідок 2.1.1. Якщо сигнатура Ω деякої алгебри G містить нульарні операції, то перетин довільної системи підалгебр алгебри G непустий.

Дійсно, якщо $\omega \in \Omega$ і $\text{ar}(\omega) = 0$, то елемент $\omega \in A$ і, значить, повинен належати всім носіям підалгебр алгебри G . Але тоді перетин цих підалгебр містить, щонайменше, елемент ω .

Із теореми 2.1.1 випливає, що коли в алгебрі G взята довільна підмножина $D \subseteq A$, то існує однозначно визначена підалгебра $\{D\}$, мінімальна серед підалгебр, які включають множину D . Це буде перетин всіх підалгебр із G , які цілком включають у себе D . Якщо $\{D\} = G$, то D називається *системою твірних* для G . Алгебра $G = \{D\}$ називається *скінченно-породженою*, якщо множина D скінчена.

П р и к л а д и 2.1.1 (скінченно-породжені алгебри)

1. Множина натуральних чисел \mathbf{N} породжується множиною $D = \{0, 1\}$ за допомогою операції додавання.

2. Множина чисел \mathbf{N}^+ породжується, очевидно, одним елементом $D = \{1\}$ за допомогою операції додавання.

3. Множина цілих чисел \mathbf{Z} породжується множиною $D = \{-1, 1\}$ за допомогою операції додавання. \blacktriangleleft

2. ВІДНОШЕННЯ КОНГРУЕНТНОСТІ

Нехай R — m -арне відношення, задане на алгебрі $G = (A, \Omega)$. Відношення R стабільне відносно n -арної операції ω , якщо для будь-яких $a_{i1}, a_{i2}, \dots, a_{im} \in A$ ($i = 1, 2, \dots, n$), таких, що $(a_{i1}, a_{i2}, \dots, a_{im}) \in R$, справедливе $(\omega(a_{11}, a_{21}, \dots, a_{n1}), \dots, \omega(a_{1m}, a_{2m}, \dots, a_{nm})) \in R$.

Відношення R стабільне на множині A , коли воно стабільне відносно кожної операції з Ω . Якщо ж R — бінарне відношення, задане на множині A , то воно називається *стабільним* відносно визначеної на цій множині n -арної операції ω , якщо для довільних елементів $a_1, a'_1, \dots, a_n, a'_n$ множини A , пов'язаних співвідношеннями

$$\begin{aligned} a_1 &R a'_1, \\ a_2 &R a'_2, \\ \dots &\dots \end{aligned}$$

$$\dots\dots\dots$$

$$a_n R a'_n,$$

справедливе співвідношення $\omega(a_1, a_2, \dots, a_n) R \omega(a'_1, a'_2, \dots, a'_n)$.

Із цих означень випливає, що totожно істинні і totожно хибні відношення на множині A стабільні відносно будь-якої основної операції, визначененої на множині A .

Відношення еквівалентності R , задане на множині A , називається **конгруентністю**, якщо R стабільне на A .

В теоремах 1.1.3, 1.1.4 розглядались деякі операції над відношеннями еквівалентності. Подивимось, які з цих операцій зберігаються для конгруентностей.

Теорема 2.1.2. Якщо R, R_i — конгруентності на множині A , то:

- 1) R^{-1} — конгруентність на множині A ;
- 2) $R \cap R_i$ — конгруентність на множині A ;
- 3) $R * R_i$ — конгруентність на множині $A \Leftrightarrow R * R_i = R_i * R$.

Доведення. 1. Якщо R — конгруентність, то за теоремою 1.1.5 $R^{-1} = R$, тобто R^{-1} — конгруентність.

2. Із теореми 1.1.5 випливає, що $R \cap R_i$ — відношення еквівалентності. Нехай $\omega \in \Omega$ ($\text{ar}(\omega) = m$) і $(a_1, a'_1) \in R \cap R_i, \dots, (a_m, a'_m) \in R \cap R_i$. Тоді за умовою теореми маємо

$(\omega(a_1, \dots, a_m), \omega(a'_1, \dots, a'_m)) \in R$ і $(\omega(a_1, \dots, a_m), \omega(a'_1, \dots, a'_m)) \in R_i$.

А звідси випливає, що $(\omega(a_1, \dots, a_m), \omega(a'_1, \dots, a'_m)) \in R \cap R_i$.

3. Із того, що відношення R і R_i можна переставляти місцями, випливає, що $R * R_i$ — відношення еквівалентності згідно з теоремою 1.1.5. Нехай $\omega \in \Omega$, $\text{ar}(\omega) = m$ і $(a_1, a'_1), \dots, (a_m, a'_m) \in R * R_i$. Тоді відповідно до означення операції добутку відношень у множині A існують елементи b_1, \dots, b_m , такі, що $(a_i, b_i) \in R$ і $(b_i, a'_i) \in R_i$, $i = 1, 2, \dots, m$. Оскільки R і R_i — конгруенції, маємо включення $(\omega(a_1, \dots, a_m), \omega(b_1, \dots, b_m)) \in R$ і $(\omega(b_1, \dots, b_m), \omega(a'_1, \dots, a'_m)) \in R_i$. Але звідси випливає, що $(\omega(a_1, \dots, a_m), \omega(a'_1, \dots, a'_m)) \in R * R_i$.

Теорема доведена.

3. ГОМОМОРФІЗМИ УНІВЕРСАЛЬНИХ АЛГЕБР

Універсальні алгебри $G = (A, \Omega)$ і $Q = (B, \Omega')$ називаються алгебрами одного **типу**, якщо між елементами сигнатур Ω і Ω' можна встановити таку взаємно однозначну відповідність, при якій всяка операція ω із Ω і відповідна їй операція ω' із Ω' будуть мати одну і ту ж арність. Отже, можна вважати, що в алгебрах одного типу задана одна і та ж сигнатура операцій.

Алгебра $G = (A, \Omega)$ називається *гомоморфною* алгебрі $Q = (B, \Omega)$ того ж типу, що і алгебра G , якщо існує відображення $h: A \rightarrow B$, таке, що для всіх елементів a_1, \dots, a_n із A і будь-якої n -арної операції ω із Ω справедлива рівність

$$h(\omega(a_1, \dots, a_n)) = \omega(h(a_1), \dots, h(a_n)). \quad (2.1.1)$$

При цьому відображення h називається *гомоморфізмом*. Якщо h — взаємно однозначне відображення алгебри G на алгебру Q , то воно називається *ізоморфізмом*, а алгебри G і Q — *ізоморфними* ($G \sim Q$). Гомоморфізм алгебри G на себе (тобто $h: G \rightarrow G$) називається *епіморфізмом*, а коли h — ізоморфізм G на G , то він називається *автоморфізмом*. Якщо алгебра G ізоморфна деякій підалгебрі алгебри G' , то говорять, що алгебра G *ізоморфно вкладається* в алгебру G' .

Позначимо $h(G)$ образ алгебри G при гомоморфізмі h алгебри G в алгебру G' . Безпосередньо з означення гомоморфізму алгебр випливають такі прості твердження.

Твердження 2.1.1. Якщо h — гомоморфізм алгебри G в алгебру G' і $h(G)$ — образ алгебри G при цьому гомоморфізмі, то $h(G)$ — підалгебра алгебри G' .

Нехай $G = (A, \Omega)$, $a_1, a_2, \dots, a_n \in A$ і $\omega \in \Omega$ — деяка операція арності n . Покажемо насамперед замкнутість множини $h(G)$ відносно основних операцій алгебри G' . Оскільки $\omega(a_1, a_2, \dots, a_n) \in G$ і h — гомоморфізм, то

$$h(\omega(a_1, \dots, a_n)) = \omega(h(a_1), \dots, h(a_n)) \in h(G),$$

бо $h(G)$ містить всі образи елементів із G . Внаслідок довільноті операції ω гомоморфний образ $h(G)$ алгебри G становить підалгебру алгебри G' .

Розглянемо конкретний випадок, коли G — алгебра, в якій виконуються такі тотожності:

- а) $a + b = b + a$;
- б) $(a + b) + c = a + (b + c)$;
- в) $a + 0 = a$;
- г) $a + (-a) = 0$.

Нехай $h: G \rightarrow G'$ — гомоморфізм, тоді маємо:

- а) $h(a + b) = h(b + a) = h(a) + h(b) = h(b) + h(a);$
- б) $h((a + b) + c) = h(a + (b + c)) = (h(a) + h(b)) + h(c) = h(a) + (h(b) + h(c));$
- в) $h(a + 0) = h(a) + h(0) = h(a);$
- г) $h(a + (-a)) = h(a) + h(-a) = h(0) = 0'$,

де $0'$ — нульовий елемент алгебри $h(G)$. Елемент $0'$ дійсно буде нульовим елементом алгебри $h(G)$ внаслідок того, що рівність

$h(a) + h(0) = h(a) + 0' = h(a)$ виконується для будь-якого a з G , і якщо a пробігає всі елементи з G , то $h(a)$ буде пробігати всі елементи з $h(G)$ і, значить, $0'$ буде нульовим елементом алгебри $h(G)$. Отже, в алгебрі, що являє собою гомоморфний образ алгебри G , теж виконуються всі закони алгебри G .

Твердження 2.1.2. *Добуток гомоморфізмів алгебр є гомоморфізмом.*

Доведення пропонується читачеві (див. вправу 1 в кінці розділу).

Існує певний загальний метод огляду всіх гомоморфних образів універсальних алгебр.

Нехай $G = (A, \Omega)$ — деяка універсальна алгебра, R — конгруенція на G і A_1, A_2, \dots, A_n — класи розбиття множини A за відношенням R . Оскільки R — конгруенція, клас B елемента $\omega(a_1, \dots, a_n)$, де $a_i \in A$, $\omega \in \Omega$, $\text{ar}(\omega) = n$, визначається однозначно і не залежить від вибору елементів a_1, a_2, \dots, a_n в класах A_1, A_2, \dots, A_n . Це дає можливість визначити операцію ω на фактор-множині G/R , якщо покласти $\omega(A_1, \dots, A_n) = B$. Оскільки сказане справедливе для будь-якої операції ω із Ω , то G/R стає універсальною алгеброю тієї ж сигнатури, що і алгебра G . Ця алгебра називається **фактор-алгеброю** алгебри G по конгруенції R . Оскільки алгебри G і G/R одного типу, то можна говорити про гомоморфізм h алгебри G на G/R , при якому кожному елементу $a \in A$ ставиться у відповідність клас еквівалентності, який містить цей елемент.

Покажемо, що таке відображення h алгебри G на алгебру G/R є гомоморфізмом.

Нехай $(a_1, a_2, \dots, a_n) \in G$, $\omega \in \Omega$, $\text{ar}(\omega) = n$. Тоді $h(\omega(a_1, a_2, \dots, a_n)) = \omega(h(a_1), h(a_2), \dots, h(a_n))$ за означенням операції ω на G/R , тобто h — гомоморфізм, який називається **натуральним гомоморфізмом**.

Теорема 2.1.3 (про гомоморфізми). Якщо f — гомоморфізм алгебри G на алгебру G' , то на алгебрі G існує така конгруентність R , що алгебра G' ізоморфна фактор-алгебрі G/R , і коли g — цей ізоморфізм, то відображення $f * g$ збігається з натуральним гомоморфізмом h алгебри G на G/R .

Д о в е д е н н я. Визначимо відношення R на класах таким чином: $(a, b) \in R \Leftrightarrow f(a) = f(b)$. Із теореми 1.1.5 випливає, що R — еквівалентність, а відображення $f * g$ взаємно однозначне. Покажемо, що R — конгруентність, а $f * g$ — ізоморфізм.

Нехай $f(a_i) = f(b_i)$ для деяких $a_i, b_i \in G$ і $\omega \in \Omega$, $\text{ar}(\omega) = n$. Тоді

$$\begin{aligned} f(\omega(a_1, a_2, \dots, a_n)) &= \omega(f(a_1), f(a_2), \dots, f(a_n)) = \\ &= \omega(f(b_1), f(b_2), \dots, f(b_n)) = f(\omega(b_1, b_2, \dots, b_n)), \end{aligned}$$

тобто R — конгруенція, і фактор-алгебра G/R існує.

Нехай a'_i — довільні елементи із G' . Задаючи $g(a'_i) = A_i$, $i = 1, 2, \dots, n$, і вибираючи елементи a_i із A_i такі, що $f(a_i) = a'_i$, одержуємо

$$f(\omega(a_1, \dots, a_n)) = \omega(f(a_1), \dots, f(a_n)) = \omega(a'_1, \dots, a'_n).$$

Оскільки $\omega(a_1, \dots, a_n) \in \omega(A_1, \dots, A_n)$, то

$$g(\omega(a'_1, \dots, a'_n)) = \omega(A_1, \dots, A_n) = \omega(g(a'_1), \dots, g(a'_n)),$$

що і потрібно було довести.

Приклад 2.1.2

Нехай $G = (\mathbf{D}^+, \Omega)$ — алгебра додатних дійсних чисел, тобто $\mathbf{D}^+ = \{x \in \mathbf{D} \mid x > 0\}$, а Ω містить бінарну операцію множення, унарну операцію взяття оберненого елемента і нульярну операцію, яка фіксує одиничний елемент. Алгебра $G' = (\mathbf{D}, \Omega')$, де \mathbf{D} — множина всіх дійсних чисел, а Ω' містить бінарну операцію додавання, унарний мінус і нульярну операцію 0.

Відображення $f = \lg$ — десятковий логарифм — є ізоморфізмом алгебри G на G' . Дійсно,

- a) $\lg(a * a') = \lg a + \lg a'$;
- б) $\lg(1/a) = \lg(1) - \lg(a) = 0 - \lg a = -\lg a$;
- в) $\lg 1 = \lg(a * (1/a)) = \lg a - \lg a = 0$.

Таким чином, операції при цьому відображенні зберігаються відповідно до означення ізоморфізму.

Покажемо тепер, що це відображення взаємно однозначне.

Нехай $a \neq a'$, але $\lg a = \lg a'$. Тоді $\lg a - \lg a' = \lg(a/a') = 0$ і, значить, $a/a' = 1$ і $a = a'$, що є хибним по відношенню до нашого припущення. Обернене твердження доводиться аналогічно.

Отже, згідно з “а”—“в” і твердженням, що відображення взаємно однозначне, отримуємо, що відображення $f = \lg$ — ізоморфізм. *

4. МОВА (АЛГЕБРА) ТЕРМІВ

Для задання функцій і операцій, а також для вивчення їх властивостей користуються особливою формальною мовою — *мовою термів*.

Загалом для задання деякої формальної мови необхідно задати її алфавіт і правила, за якими будуються слова із символів цього алфавіту. Довільна сукупність попарно різних символів $X = \{x_1, x_2, \dots, x_n, \dots\}$ називається *алфавітом*. Символи алфавіту часто ще називають *буквами*. Найпростіший приклад мови — це мова слів в алфавіті X . Із скінченної послідовності $p = x_{i1} \dots x_{in}$

букв складається слово в алфавіті X , якщо $x_{ij} \in X$, $j = 1, 2, \dots, k$. При цьому букви x_{ij} називаються буквами, з яких складається слово p , а їх число — довжиною $l(p)$ слова p . Крім слів, довжина яких виражається цілим додатним числом, розглядається слово нульової довжини, яке за означенням не має жодного символу і називається *пустим словом*. Для позначення цього слова вводиться спеціальний символ $e \in X$.

Зауважимо, що коли p, q — деякі слова в алфавіті X , тобто послідовності $x_{i1} \dots x_{ik}$ та $x_{j1} \dots x_{jl}$ відповідно, то послідовність $pq = x_{i1} \dots x_{ik}x_{j1} \dots x_{jl}$, одержана в результаті приписування слова q безпосередньо після останнього символу слова p , теж буде, очевидно, словом в алфавіті X . Це випливає безпосередньо з означення слова. Отже, таке сполучення слів в алфавіті X можна розглядати як операцію на множині слів алфавіту X . Ця операція має назву *конкатенація* або *дубуток слів*.

Умовимося позначати через $F(X)$ множину всіх слів в алфавіті X , включаючи і пусте слово. Множину $F(X)$ будемо називати *мовою слів*.

Означення мови термів дається за допомогою індукції таким чином.

Алфавіт мови термів складається із символів, розбитих на три групи: T_0 , F і $\{(,)\}$. Символи з T_0 називаються *предметними*. Такими символами служать букви a, b, x, y, \dots або ті ж самі букви з індексами. Символи з F називаються *функціональними*. Це букви з верхніми і, можливо, нижніми індексами: f_2^3, g_2, f_5^5, \dots Верхній індекс n ($n \geq 1$) вказує (як і раніше) на арність функціонального символу. Якщо його немає, то функціональний символ вважається унарним. Символи третьої групи — це *ліва і права круглі дужки та кома*.

Термами називаються слова, побудовані за такими правилами:

- 1) всі символи з T_0 — терми;
- 2) якщо t_1, \dots, t_n — терми, то слово $f^n(t_1, \dots, t_n)$ — терм ($f^n \in F$, $n \geq 1$);
- 3) термами є ті і тільки ті слова, про які йде мова в пп. 1, 2.

Множина всіх термів в алфавіті X сигнатури Ω позначається $T(\Omega, X)$. Якщо $t = \omega(t_1, \dots, t_n) \in T(\Omega, X)$, то терми t_1, \dots, t_n називаються *безпосередніми підтермами терма* t . Транзитивне замикання відношення *безпосередній підтерм* називається *відношенням підтерм терма*.

Перейдемо тепер до означення вільної алгебри заданого класу алгебр.

Множину термів $T(\Omega, X)$ можна розглядати як універсальну Ω -алгебру, якщо визначитися з операціями нульової арності, ос-

кільки при визначенні $T(\Omega, X)$ вважалося, що $\text{аг}(\omega) \geq 1$ для кожної операції $\omega \in \Omega$.

Нехай $X = \{x_1, x_2, \dots, x_n, \dots\}$ — деякий алфавіт, а Ω — множина операцій. Представимо множину Ω у вигляді $\Omega = \Omega' \cup \Omega_0$, де Ω_0 означає множину нульлярних операцій, а $\Omega' = \Omega \setminus \Omega_0$. Позначимо $T(\Omega', X)$ множину термів, в якої $T_0 = X \cup \Omega_0$, а $F = \Omega'$. Тепер $T(\Omega', X)$ можна розглядати як універсальну Ω' -алгебру з системою операцій Ω' . Надалі, щоб не ускладнювати викладки і не вводити зайві позначення, алгебру $T(\Omega', X)$ будемо позначати, як і раніше, $T(\Omega, X)$, маючи на увазі описаний перехід, і називати її **алгеброю термів**.

Нехай маємо довільну алгебру $G = (A, \Omega)$ (алгебра G може збігатися з $T(\Omega, X)$) і алгебру $T(\Omega, X)$. Якщо $\Omega_0 \neq \emptyset$, то нехай $a(f)$ означає елемент із A , який відповідає нульлярній операції f із Ω_0 . Розглянемо відображення $h: T_0 \rightarrow A$, таке, що $h(f) = a(f)$ для $f \in \Omega_0$. Відображення h можна продовжити на всю алгебру $T(\Omega, X)$, якщо для $p_1, p_2, \dots, p_n \in T(\Omega, X)$ і $f^n \in \Omega$ ($n \geq 1$) покласти $h(f(p_1, \dots, p_n)) = f(h(p_1), \dots, h(p_n))$.

Відображення h називається **інтерпретацією** $T(\Omega, X)$ на G .

Говорять, що в алгебрі G виконується **тотожне співвідношення** (або просто **тотожність**) $p_1 = p_2$, коли $h(p_1) = h(p_2)$ в G при будь-якій інтерпретації h . Якщо дана множина тотожних співвідношень Eq , то з сукупності алгебр сигнатури Ω , в яких виконуються всі тотожні співвідношення з Eq , складається клас алгебр, який позначатиметься $K(\Omega, Eq)$. Співвідношення з множини Eq називаються **тотожними** або **тотожностями**.

Нехай $p_1 = p_2 \in Eq$, $p, q \in T(\Omega, X)$ і p одержане з q внаслідок підстановки p_1 замість деякого входження терма p_2 в терм q . У цьому випадку говорять, що терм p **безпосередньо виводиться** з терма q за допомогою тотожного співвідношення $p_1 = p_2$ із Eq . Транзитивне замикання відношення безпосереднього виведення називається просто **виведенням**. Терми q_1 і q_2 називаються **еквівалентними** відносно Eq , якщо один із них виводиться з другого ($p_1 Req q_2$). Відношення Req буде, очевидно, **еквівалентністю** і нарешті, як легко перевірити, **конгруентністю**.

Нехай $T(\Omega, X, Eq)$ означає фактор-алгебру $T(\Omega, X) / Req$. Алгебра $T(\Omega, X, Eq)$ називається **вільною алгеброю класу** $K(\Omega, Eq)$, а множина X — її **системою вільних твірних (базисом)**. Із означення $T(\Omega, X, Eq)$ випливає, що $T(\Omega, X, Eq) \in K(\Omega, Eq)$. Важливість поняття вільної алгебри випливає з такої теореми.

Теорема 2.1.4. *Кожна алгебра G із класу $K(\Omega, Eq)$ є гомоморфним образом вільної алгебри $T(\Omega, X, Eq)$ цього класу [41, 42].*

Алгебра $T(\Omega, X, Eq)$ називається *спадково вільною*, якщо всяка її підалгебра — вільна алгебра.

5. ПОХІДНІ ОПЕРАЦІЇ І СКІНЧЕННІ АЛГЕБРИ

Крім основних операцій алгебри, часто розглядаються похідні операції, число яких може бути нескінченим.

Нехай $\omega(x_1, \dots, x_n) \in T(\Omega, X, Eq)$, $\text{ar}(\omega) > 1$ і $x_1, \dots, x_n \in X$ — вільні твірні. Замінимо k ($0 \leq k \leq n$) вільних твірних, наприклад x_{n-k+1}, \dots, x_n , деякими фіксованими елементами a_1, a_2, \dots, a_k із $T(\Omega, X, Eq)$. Вираз $\omega(x_1, \dots, x_{n-k}, a_1, \dots, a_k)$, який при цьому буде одержаний, визначає на $T(\Omega, X, Eq)$ ($n - k$)-арну операцію: системі елементів $b_1, \dots, b_{n-k} \in T(\Omega, X, Eq)$ ця операція ставить у відповідність однозначно визначений елемент $\omega(b_1, \dots, b_{n-k}, a_1, \dots, a_k)$.

Всі операції, які можуть бути одержані в $T(\Omega, X, Eq)$ таким чином, називаються *похідними операціями алгебри* $T(\Omega, X, Eq)$. Похідна операція називається *головною*, якщо $k = 0$, тобто коли в термі $\omega(x_1, \dots, x_n)$ не виконується жодної заміни вільних твірних. Зауважимо, що коли сигнатура Ω містить похідні операції, то таке поняття, як підалгебра даної алгебри втрачає сенс, оскільки підалгебра не обов'язково повинна містити елементи a_1, \dots, a_k , які беруть участь в означенні похідної операції. Analogічне зауваження стосується і поняття гомоморфізму. Всі ці складноші зникають, якщо вважати, що кожна операція сигнатури Ω — головна.

Надалі приймемо, що до складу сигнатури Ω входять тільки головні операції і не входять похідні операції.

На завершення цього підрозділу коротко зупинимося на скінченних алгебрах.

Алгебра називається *скінченною*, якщо її носій має скінченне число елементів.

Якщо алгебра G скінчenna і складається з n елементів, то її називають *алгеброю порядку n* .

Скінченну алгебру G в деяких випадках зручно задавати у вигляді прямокутних таблиць, які називаються *таблицями Келі*. Кожна така таблиця відповідає деякій операції ω із Ω і буде відповідати за такими правилами.

Якщо ω — k -арна операція з Ω і $\omega(a_1, a_2, \dots, a_k) = b$, то в таблиці цьому виразу відповідає рядок a_1, a_2, \dots, a_k, b . Іншими словами, кожній операції ω алгебри G відповідає рядок елементів a_1, a_2, \dots, a_k, b , такий, що $\omega(a_1, a_2, \dots, a_k) = b$.

ω	1	2	...	k	$k+1$
1	a_{11}	a_{12}	...	a_{1k}	b_1
2	a_{21}	a_{22}	...	a_{2k}	b_2
...
q	a_{q1}	a_{q2}	...	a_{qk}	b_q

де $q \leq n$.

Якщо ж ω — бінарна операція, то ця таблиця задається квадратною таблицею

ω	a_1	a_2	...	a_n
a_1	a_{11}	a_{12}	...	a_{1n}
a_2	a_{21}	a_{22}	...	a_{2n}
...
a_n	a_{n1}	a_{n2}	...	a_{nn}

в якій на перетині i -го рядка і j -го стовпчика стоїть значення $\omega(a_i, a_j)$. Коли операція унарна, то це буде просто рядок, який складається з n елементів, i -й елемент якого дорівнює значенню $\omega(a_i)$.

Зauważимо, що такого роду таблиці можна задавати більш звичними числовими таблицями, використовуючи відображення $h(a_i) = i$.

Приклади відповідних представлень скінчених алгебр будуть розгляdatися детальніше в наступних розділах, а тепер перейдемо до розгляду основних добре відомих алгебр, які відіграють важливу роль як в теорії, так і на практиці.

§ 2.2. ВІЛЬНІ АЛГЕБРИ ТА ЇХ ОСНОВНІ ВЛАСТИВОСТІ

1. АБСОЛЮТНО ВІЛЬНІ АЛГЕБРИ

Алгебра $T(\Omega, X, Eq)$ називається *абсолютно вільною*, коли $T(\Omega, X, Eq) = T(\Omega, X)$, тобто тоді і тільки тоді, коли $Eq = \emptyset$.

Із означення алгебри термів $T(\Omega, X)$ випливає, що множина X служить для цієї алгебри системою твірних і що алгебри термів над рівнопотужними алфавітами ізоморфні. Оскільки в $T(\Omega, X)$ не виконується жодне нетривіальне співвідношення, то всякий елемент із $T(\Omega, X)$ єдиним способом виражається через елементи алфавіту X і символи нульарних операцій із Ω .

Зміст терміна *абсолютно вільна* для алгебри $T(\Omega, X)$ в класі всіх алгебр сигнатури Ω випливає з такого твердження.

Твердження 2.2.1. *Всяке відображення f множини твірних X алгебри термів $T(\Omega, X)$ в будь-яку алгебру G того ж типу, що й ал-*

гебра $T(\Omega, X)$, можна продовжити єдиним способом до гомоморфізму h алгебри $T(\Omega, X)$ в алгебру G .

Дійсно, покладемо $h(x) = f(x)$ для всіх x із X і $h(\omega) = \omega$ для всіх нульарних операцій ω із Ω . Якщо для $\omega \in \Omega$, $\text{аг}(\omega) = n \geq 1$, образи термів t_1, t_2, \dots, t_n уже визначені, то, покладаючи $h(\omega(t_1, t_2, \dots, t_n)) = \omega(h(t_1), h(t_2), \dots, h(t_n))$, одержуємо, що відображення h — гомоморфізм.

Єдиність такого відображення h випливає з єдності представлення термів в алгебрі $T(\Omega, X)$.

2. ВІЛЬНИЙ ГРУПОЇД

Абсолютно вільна алгебра $T(\Omega, X)$ називається вільним *групоїдом*, якщо Ω містить єдину бінарну операцію, яка називається *множенням слів в алфавіті* X . Словом у цій алгебрі буде всяка скінченна упорядкована система елементів із $X x_{i1} x_{i2} \dots x_{in}$ ($n > 1$), де x_{ij} ($j = 1, 2, \dots, n$) не обов'язково всі різні, причому в цій алгебрі заданий розподіл дужок, які визначають порядок виконання операції множення. При цьому кожний символ $x_{ij} \in X$ вважається взятым в дужки, а множення двох слів означає, що задані слова беруться в дужки і пишуться одне за другим. Наприклад, якщо

$$p = (x_{i1})(x_{i2}), q = (x_{i1})((x_{i2})(x_{i1})), \text{ то } pq = ((x_{i1})(x_{i2}))((x_{i1})((x_{i2})(x_{i1}))).$$

3. ВІЛЬНА НАПІВГРУПА

Якщо Ω містить єдину бінарну операцію — множення, а Eq — єдине тотожне співвідношення — асоціативність множення, то $T(\Omega, X, Eq)$ називається *вільною напівгрупою*. Коли існує елемент $e \in T(\Omega, X, Eq)$ і Eq містить тотожності $re = er = r$ для всіх $r \in T(\Omega, X, Eq)$, то $T(\Omega, X, Eq)$ називається *вільною напівгрупою з одиницею* або *вільним моноїдом*.

Приклади 2.2.1 (напівгруп)

1. Прикладом вільної напівгрупи $T(\Omega, X)$ може служити введена раніше мова $F(X)$ — мова слів в алфавіті X .

Нехай $p, q \in T(\Omega, X)$. Роль операції множення в цій напівгрупі відіграє операція конкатенації, яка, як легко зрозуміти, задовольняє закон асоціативності. Якщо пусте слово e належить напівгрупі $T(\Omega, X)$, то ця напівгрупа буде мати одиницю. Підтерм p терма q називається *підсловом* слова q . Якщо p — підслово слова q , то q можна представити у вигляді добутку

$$q = p'pp'', \quad (2.2.1)$$

де p' , p'' — підходячі (можливо пусті) слова з $T(\Omega, X)$. Якщо в розкладі (2.2.1) слово p' має найменшу можливу довжину, то говорять про перше входження слова p в слово q . Аналогічно можна говорити про друге, третє і т.д. входження p в q . Нехай розклад (2.2.1) відповідає k -му входженню p в q . Тоді слово $q = p'q'p''$ називається словом, одержаним в результаті підстановки слова q' замість k -го входження p в q .

Твердження 2.2.2. *Множина слів мови $F(X)$ над скінченним алфавітом X є зліченою множиною.*

Доведення випливає з того, що цю мову можна подати у вигляді об'єднання

$$F(X) = \{e\} \cup L_1 \cup L_2 \cup \dots \cup L_n \cup \dots,$$

де L_k — множина всіх елементів із $F(X)$, довжина яких дорівнює k , $k = 1, 2, \dots$ Теорема 1.1.18 підтверджує справедливість даного твердження.

Твердження 2.2.3. *Множина всіх мов над скінченним алфавітом X є незліченою множиною [19].*

Дійсно, множина всіх мов в алфавіті X є булевом множини X , і справедливість даного твердження випливає тепер із теореми про булеван.

2. Напівгрупою є сукупність всіх бінарних відношень, заданих на деякій множині A , щодо операції множення відношень, оскільки ця операція асоціативна (теорема 1.1.3). Це буде часткова напівгрупа з одиницею. Роль одиниці відіграє відношення тотожності i_A .

Сукупність всіх відношень еквівалентності, заданих на множині A , не буде напівгрупою, оскільки ця сукупність у загальному випадку не замкнута відносно операції множення (теорема 1.1.5). З цієї причини і сукупність всіх конгруентностей на множині A не буде напівгрупою.

3. Перетворенням множини A називається всяке відображення f множини A в себе, тобто $f: A \rightarrow A$. Очевидно, що коли f і g — перетворення множини A , то $f * g$ — теж перетворення множини (див. вправи), тобто множина всіх перетворень множини A замкнута відносно операції множення перетворень. Як було показано вище, операція множення відображень асоціативна, і, значить, сукупність всіх перетворень множини A становить напівгрупу, яка називається *симетричною напівгрупою* на множині A . Важливість цієї напівгрупи висвітлює така теорема.

Теорема 2.2.1. *Всяка напівгрупа G ізоморфно вкладається в симетричну напівгрупу на деякій множині A [42].*

Якщо множина $A = \{1, 2, \dots, n\}$ скінчена, то симетрична напівгрупа на множині A теж скінчена. У цьому випадку всяке перетворення f задається таблицею

$$f = \begin{pmatrix} 1 & 2 & \dots & n \\ a_1 & a_2 & \dots & a_n \end{pmatrix},$$

де кожний стовпчик означає $f(i) = a_i$, $a_i \in A$.

4. Нехай G — напівгрупа і a — довільний її елемент. Асоціативність множення дозволяє звичайним способом визначити додатні степені a^n елемента a , $n = 1, 2, \dots$, причому $a^k \cdot a^l = a^{k+l}$, $(a^k)^l = a^{kl}$. Звідси видно, що додатні степені елемента a становлять піднапівгрупу напівгрупи G . Ця піднапівгрупа комутативна і називається *циклічною піднапівгрупою* елемента a . \blacktriangleleft

4. ВІЛЬНА КОМУТАТИВНА НАПІВГРУПА

Вільна напівгрупа $T(\Omega, X, Eq)$ називається *вільною комутативною напівгрупою*, якщо множина Eq , крім асоціативності, містить токожне співвідношення комутативності множення слів.

Тотожні співвідношення дозволяють записати будь-яке слово із $T(\Omega, X, Eq)$ у вигляді $x_1^{n_1} x_2^{n_2} \dots x_k^{n_k}$, де $x_j \in X$, n_j — цілі невід'ємні числа. Під виразом x^n розуміють слово $xx\dots x$ довжини n при $n > 0$, а при $n = 0$ — $x^n = e$.

Прикладом вільної комутативної напівгрупи відносно операції додавання може служити множина \mathbb{N}^+ , яка, як було сказано вище, породжується єдиною вільною твірною — 1, тобто $X = \{1\}$. Операція додавання на \mathbb{N}^+ асоціативно-комутативна.

5. ВІЛЬНІ ГРУПИ

Нехай Ω містить бінарну операцію конкатенації слів, унарну операцію взяття оберненого елемента і нульарну операцію, яка фіксує одиничний елемент. Наявність операції взяття оберненого елемента дозволяє однозначно поставити у відповідність множині X множину елементів X^{-1} . Символи x із X і x^{-1} із X^{-1} називаються *взаємно оберненими*. Слово в алфавіті $\bar{X} = X \cup X^{-1}$ називається *некорочуваним*, якщо воно не має жодної пари взаємно обернених символів, які стоять поруч. Довільне слово можна петретворити в некорочуване слово внаслідок послідовного викреслювання пар взаємно обернених символів, які стоять поруч.

Очевидно, що пусте слово e буде нескорочуваним. Таке перетворення слів називається їх *приведенням*.

Якщо Eq містить співвідношення асоціативності, співвідношення скорочення ($x \cdot x^{-1} = x^{-1} \cdot x = e$) і співвідношення, пов'язані з одиницею ($x \cdot e = e \cdot x = x$), то $T(\Omega, X, Eq)$ називається *вільною групою*. Одиницею групи $T(\Omega, X, Eq)$ є пусте слово e . Тотожні співвідношення дозволяють записати довільне слово з $T(\Omega, X, Eq)$ у вигляді нескорочуваного слова $x_{i_1}^{n_1} x_{i_2}^{n_2} \dots x_{i_k}^{n_k}$, де x_{i_j} із \bar{X} , n_j — ціле число, x_{i_j} не обов'язково всі різні, $j = 1, \dots, k$, а

$$x^n = \begin{cases} xx\dots x, & \text{коли } n > 0, \\ e, & \text{коли } n = 0, \\ x^{-1}x^{-1}\dots x^{-1}, & \text{коли } n < 0. \end{cases}$$

Нехай G — група, H — деяка її підгрупа і a — довільний елемент із G . Множина $aH = \{ah \mid h \in H\}$ називається *лівим суміжним класом* групи G по підгрупі H , заданим елементом a . Ясно, що $a \in aH$, оскільки $e \in H$, і якщо $b \in aH$, то $bH = aH$. Дійсно, $b \in aH$ означає, що $b = ah$, де $h \in H$. Але тоді для будь-яких $h_1, h_2 \in H$, маємо $bh_1 = (ah)h_1 = a(hh_1) \in aH$, тобто $bH \subseteq aH$. З іншого боку, $ah_2 = (bh^{-1})h_2 = b(h^{-1}h_2) \in bH$, тобто $aH \subseteq bH$. Значить, $aH = bH$.

Звідси випливає, що ліві суміжні класи $H, a_1H, a_2H, \dots, a_nH$ — це класи розбиття групи G . За теоремою 1.1.4 підгрупа H задає на G відношення еквівалентності R : $aRb \Leftrightarrow aH = bH$.

Задання групи G у вигляді об'єднання класів $H \cup a_1H \cup a_2H \cup a_3H \cup \dots \cup a_nH \cup \dots$ називається *лівостороннім розкладом* групи G по підгрупі H . Аналогічно будеться і правосторонній розклад групи G по підгрупі H . Правосторонній і лівосторонній розклади групи складаються з одного і того ж числа класів. У цьому легко переконатися, задаючи відображення $f: G \rightarrow G$ так, що $f(a) = a^{-1}$.

Якщо ж число суміжних класів у розкладі групи G по підгрупі H скінченне, то підгрупа H називається підгрупою *скінченного індексу*, а число класів — *індексом* підгрупи H в групі G .

Підгрупа H називається *нормальним дільником* або *інваріантною підгрупою* групи G , якщо лівосторонній розклад G по H збігається з правостороннім розкладом G по H . Інакше кажучи, для всякого $a \in G$ виконується рівність $aH = Ha$. Очевидно, що нормальними дільниками всякої групи буде однічна підгрупа, яка складається лише з одиниці групи, і сама група. Ці нормальні дільники називаються *тривіальними*.

Нехай G і G' — групи, e і e' — їх одиничні елементи відповідно, а h — гомоморфізм групи G в групу G' . Множина $\{a \in G \mid h(a) = e'\}$ називається **ядром гомоморфізму** h і позначається символом $\ker(h)$. Неважко довести, що $\ker(h)$ — нормальний дільник групи G (див. вправу 15).

Якщо група скінчена, то всі її підгрупи теж будуть скінченими, а число її елементів називатиметься **порядком групи**. Нехай G — скінчена група порядку n і H — її підгрупа порядку k . Тоді при розкладі G по H всякий суміжний клас складається рівно з k елементів і, значить, $n = kj$. Звідси випливає така теорема.

Теорема Лагранжа. *Порядок i індекс будь-якої підгрупи скінченної групи ϵ дільниками порядку групи.*

Якщо G — група і $a \in G$, то візьмемо $a^0 = e$ і $a^{-n} = (a^{-1})^n$, а додатні степені елемента a було визначено вище для циклічної напівгрупи елемента a .

Сукупність елементів a^n , де $n \in \mathbb{Z}$, називається **циклічною підгрупою** групи G . Степені елемента a не обов'язково мають бути різними. Елемент a групи G називається **елементом скінченного порядку**, якщо існують такі цілі числа k і l , що $a^k = a^l$, тобто $a^{k-l} = e$. Найменший показник серед усіх таких показників елемента a називається **порядком елемента** a . Якщо таких чисел k і l не існує, то a має нескінчений порядок.

Група, всі елементи якої мають нескінчений порядок, називається **групою без кручень**. Група, всі елементи якої мають скінченні порядки (не обов'язково обмежені в сукупності), називається **періодичною групою**.

Якщо a — елемент скінченного порядку, то $a^0 = e$, a , a^2 , ..., a^{n-1} будуть різними елементами групи. Якщо дано a^k , де $k > n$, то $k = qn + r$, $0 \leq r < n$, і $a^{qn+r} = (a^n)^q \cdot a^r = a^r$. Значить, порядок елемента скінченного порядку збігається з порядком його циклічної групи.

Наслідок 1 теореми Лагранжа. Порядок будь-якого елемента скінченної групи є дільником порядку групи.

Наслідок 2 теореми Лагранжа. Всяка скінчена група, порядок якої — просте число, буде циклічною.

Дійсно, внаслідок простоти порядку групи вона повинна збігатися з циклічною підгрупою, породженою будь-яким її елементом, що не дорівнює одиниці.

Група називається **простою**, якщо вона не має нетривіальних нормальніх дільників.

Якщо група H — нормальний дільник групи G , то множина суміжних класів становить групу. Дійсно, якщо H, a_1H, a_2H, \dots — розбиття групи G , і для довільного елемента a із G виконуються рівності $aH = Ha$ і $H * H = H$, то неважко довести, що:

- 1) $(aH * bH) * cH = aH * (bH * cH);$
- 2) $aH * H = H * aH = aH = Ha$, тобто H грає роль одиниці;
- 3) $aH * (a^{-1})H = (aH * a^{-1}) * H = H * H = H;$
- 4) $aH * bH = (Ha) * (bH) = (H(ab)) * H = abH * H = Hab.$

Інакше кажучи, всякий нормальнй дльник H групи G задає деяке відношення конгруентності R на G . Фактор-група за цим відношенням конгруентності позначається G/H і називається **фактор-групою** групи G за нормальним дльником H .

Якщо група G скінчена і H — її нормальний дльник, то з цього випливає такий наслідок.

Наслідок 3 теореми Лагранжа. Порядок групи G/H дорівнює індексу групи H в групі G і є дльником порядку групи G .

П р и к л а д 2.2.2

Розглянемо приклад некомутативної групи, яка грає важливу роль в теорії груп, — *групу підстановок деякої скінченної множини $M = \{1, 2, \dots, n\}$* . Вище було введено поняття перетворення множини (див. приклади напівгруп). Окремим випадком перетворення є *підстановка*, тобто взаємно однозначне відображення множини M на себе.

Ми вже знаємо, що добуток відображень — це їх послідовне виконання і що ця операція асоціативна.

Сукупність всіх підстановок множини M становить групу. Дійсно, роль одиниці в цій групі грає тотожна підстановка, яка залишає на місці кожний елемент із M . З іншого боку, якщо елемент a переходить в елемент $f(a)$, то $f^{-1}(f(a)) = a$ і f^{-1} теж буде підстановкою внаслідок взаємної однозначності f , а f^{-1} буде грати роль оберненого елемента для f . Отже, всі підстановки множини M становлять групу, яка називається *симетричною групою на множині M* .

Якщо множина M скінчена і складається з n елементів, то симетрична група на M , яка називається *симетричною групою n -го степеня*, буде скінченою і матиме порядок $n!$ (див. вправу 14). Підстановки часто задаються у вигляді таблиць відповідностей.

Наприклад, якщо $M = \{1, 2, 3, 4, 5, 6\}$, то підстановка f множини M , що дорівнює $\{4, 3, 1, 5, 2, 6\}$, матиме вигляд

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 3 & 1 & 5 & 2 & 6 \end{pmatrix}$$

Множення підстановок виконується таким чином:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ \downarrow \\ 4 & 3 & 1 & 5 & 2 & 6 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ \downarrow \\ 6 & 3 & 5 & 2 & 4 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ \downarrow \\ 2 & 5 & 6 & 4 & 3 & 1 \end{pmatrix}$$

Нехай маємо три підстановки: f_1, f_2, f_3 , які задані такими таблицями:

$$f_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 3 & 5 & 2 & 6 & 1 \end{pmatrix}, f_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 5 & 2 & 6 & 1 & 4 \end{pmatrix},$$

$$f_3 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 3 & 5 & 1 & 4 & 2 \end{pmatrix}.$$

Тоді

$$(f_1 \cdot f_2) \cdot f_3 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 6 & 4 & 1 & 5 \end{pmatrix}, f_1 \cdot (f_2 \cdot f_3) = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 6 & 4 & 1 & 5 \end{pmatrix}.$$

Розглянемо приклад задання групи G всіх підстановок множини $M = \{1, 2, 3\}$ за допомогою таблиці операцій. У цьому випадку група G є скінчена алгебра порядку 6 і її елементами є підстановки:

$$f_1 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}, f_2 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}, f_3 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix},$$

$$f_4 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}, f_5 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}, f_6 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}.$$

Таблицю множення цієї алгебри, де $h(f_i) = i$, запишемо у вигляді

.	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	3	1	6	4	5
3	3	1	2	5	6	4
4	4	5	6	1	2	3
5	5	6	4	3	1	2
6	6	4	5	2	3	1

Таблиця операції взяття оберненого елемента наведена в таблиці множення, тому приводити її немає сенсу. ▲

6. ВІЛЬНІ АБЕЛЕВІ ГРУПИ

Вільна група $T(\Omega, X, Eq)$ називається *вільного абелевою групою*, якщо, крім тотожних співвідношень, які визначають її як вільну

групу, множина Eq містить тотожне співвідношення комутативності. Всяке слово з $T(\Omega, X, Eq)$ можна записати так:

$$x_1^{n_1} x_2^{n_2} \dots x_k^{n_k}, \quad (2.2.2)$$

де $x_j \in X$, x_j всі попарно різні; n_j — цілі числа, $j = 1, 2, \dots, k$. Запис слова (2.2.2) називають **мультиплікативним**, але часто для запису слів абелевої групи (як і слів комутативної напівгрупи) використовують адитивний запис — у вигляді суми

$$n_1 \cdot x_1 + n_2 \cdot x_2 + \dots + n_k \cdot x_k. \quad (2.2.3)$$

Числа n_j в цьому виразі називаються **коєфіцієнтами**. Додавання слів за формулою (2.2.3) визначається як додавання коєфіцієнтів при одинакових елементах $x_i \in X$, а роль одиниці при такій формі запису відіграє нуль — нульовий елемент, тобто вираз (2.2.3), у якого всі коєфіцієнти дорівнюють нулю. Якщо деякі x_j із X у виразі (2.2.3) відсутні, то вважається, що коєфіцієнти n_j при них дорівнюють нулю.

Приклади 2.2.3 (абелевих груп)

1. Множина $M = \{-1, 1\}$ являє собою групу відносно звичайного множення чисел. Дійсно, $1 \cdot 1 = 1$, $(-1) \cdot 1 = 1 \cdot (-1) = -1$, $(-1) \cdot (-1) = 1$. Отже, множина M замкнута відносно операції множення. Роль одиниці виконує елемент 1. Ця множина також замкнута і відносно операції взяття оберненого елемента (елементи 1 і -1 взаємно обернені). Очевидно, що операція множення асоціативна і комутативна.

2. Множина $M = \{0, 1\}$ є абелева група відносно операції додавання, якщо покласти $1 + 1 = 0$ (група лишків по модулю 2). Дійсно, $0 + 0 = 0$, $0 + 1 = 1 + 0 = 1$, $1 + 1 = 0$ і, значить, M замкнута відносно операції додавання і взяття оберненого елемента (1 є елементом, оберненим до самого себе). Одиничним елементом служить 0. Очевидно, що операція додавання асоціативно-комутативна.

Дану конструкцію групи можна узагальнити. Нехай $\mathbf{Z}(n) = \{0, 1, \dots, n-1\}$, $n > 1$. Задамо на множині $\mathbf{Z}(n)$ операцію додавання \oplus :

$$k \oplus l = \begin{cases} k + l, & \text{коли } k + l < n, \\ k + l - n, & \text{коли } k + l \geq n, \end{cases}$$

де $+ i$ — звичайні операції додавання і віднімання цілих чисел. Множина $\mathbf{Z}(n)$ з заданою таким чином операцією додавання є абелева група порядку n . Дійсно, роль нуля відіграє елемент 0, роль оберненого елемента до даного елемента k — елемент l ,

такий, що $k + l = n$. Очевидно, що операція додавання комутативна внаслідок комутативності операції додавання цілих чисел. Доведення асоціативності операції пропонується як проста вправа.

Група $Z(n)$ називається *групою лішків по модулю n* і виникає внаслідок разбиття множини цілих чисел Z на класи, де в один клас попадають ті і тільки ті числа, які при діленні на $n > 1$ дають однакові остачі. Якщо позначити $\text{rest}(m, n)$ остачу від ділення числа m на n , то операцію додавання \oplus можна визначити іншим способом:

$$k \oplus l = \text{rest}(k + l, n).$$

Очевидно, що розглянута вище група $M = \{0, 1\}$ збігається з групою $Z(2)$.

3. Множина цілих чисел Z є адитивна група, але Z не є групою відносно операції множення, оскільки операція ділення в множині Z (операція взяття оберненого) не завжди визначена.

Множина Z_2 парних чисел являє собою адитивну групу і є підгрупою групи Z . Взагалі, адитивною групою буде всяка множина цілих чисел, кратних деякому заданому цілому числу n .

Множина непарних чисел не буде групою, оскільки вона не замкнута відносно операції додавання.

4. Множина раціональних чисел RC відносно додавання є групою, але відносно множення RC не буде групою, оскільки ділення на 0 неможливе. Зауважимо, що $RC \setminus \{0\}$ — мультиплікативна група раціональних чисел.

5. Циклічна група G , тобто група, породжена одним елементом, наприклад a , складається з елементів a^n , де $n \in Z$, $a^0 = e$, $a^1 = a$. Множення елементів визначається як додавання степенів, тобто

$$a^k \cdot a^l = a^{k+l}. \quad \blacktriangleleft$$

Приклад 2.2.4 (гомоморфізм у груп)

Адитивна група цілих чисел Z гомоморфно відображається на мультиплікативну групу M (див. 2.2.3. (приклади абелевих груп), приклад 1)). Відображення h задається таким чином:

$$h(n) = \begin{cases} 1, & \text{коли } n \text{ парне,} \\ -1, & \text{коли } n \text{ непарне.} \end{cases}$$

Покажемо, що h — гомоморфізм. Спочатку покажемо, що $h(m + n) = h(m) \cdot h(n)$. Можливі такі чотири випадки:

- | | |
|--------------------------|------------------------------|
| a) $m = 2k, n = 2l;$ | b) $m = 2k, n = 2l + 1;$ |
| в) $m = 2k + 1, n = 2l;$ | г) $m = 2k + 1, n = 2l + 1.$ |

У випадку “а” маємо $h(m + n) = h(2k + 2l) = h(2(k + l)) = 1$. З іншого боку, $h(m)h(n) = h(2k)h(2l) = 1 \cdot 1 = 1$.

У випадку “б” маємо $h(m + n) = h(2(k + l) + 1) = -1$. З іншого боку, $h(m) \cdot h(n) = h(2k) \cdot h(2l + 1) = 1 \cdot (-1) = -1$.

Випадок “в” аналогічний випадку “б”.

У випадку “г” маємо $h(m + n) = h(2k + 1)h(2l + 1) = h(2(k + l + 1)) = 1$. З іншого боку, $h(m) \cdot h(n) = h(2k + 1) \cdot h(2l + 1) = -1 \cdot (-1) = 1$.

Отже, рівність $h(m + n) = h(m) \cdot h(n)$ виконується. Далі, оскільки $h(0) = 1$, то нульарні операції теж відповідають цій рівності. І останнє, якщо

$$h(n) = \begin{cases} 1, & \text{коли } n \text{ парне,} \\ -1, & \text{коли } n \text{ непарне,} \end{cases}$$

тоді

$$h(-n) = \begin{cases} 1, & \text{коли } n \text{ парне,} \\ -1, & \text{коли } n \text{ непарне.} \end{cases}$$

Оскільки елементи 1 і -1 взаємно обернені, то і в цьому випадку все коректно.

Очевидно, що це відображення не взаємно однозначне.

Таким чином, побудоване відображення h — гомоморфізм. ◀

Приклад 2.2.5 (ізоморфізм у групах)

Теорема 2.2.2. Всяка нескінчена циклічна група ізоморфна адитивній групі цілих чисел \mathbf{Z} . Всяка скінчена циклічна група порядку n ізоморфна групі лішків $\mathbf{Z}(n)$.

У першому випадку відображення задається таким чином: $h(a^n) = n$, а в другому — $h(a^k) = \text{rest}(k, n)$.

Доведення. Покажемо, що h — ізоморфізм для першого випадку:

a1) $h(a^m \cdot a^n) = h(a^{m+n}) = m + n$;

a2) $h(a^{-n}) = -n$, і оскільки a^n обернений до a^{-n} , то умови ізоморфізму виконуються;

a3) $h(a^0) = h(e) = 0$.

Залишається показати, що h взаємно однозначне. Для цього необхідно зауважити, що $a^m = a^n \Rightarrow a^{m-n} = a^0 = e$, тобто $m = n$.

Це і потрібно було довести.

Другий випадок пропонується як вправа. ◀

Приклад 2.2.6 (побудови "нетрадиційної" арифметики на основі абелевих груп)

Нехай задана деяка скінчена множина цілих чисел, наприклад $N5 = \{0, 1, 2, 3, 4\}$. Якщо ми хочемо побудувати адитивну абелеву групу, то ця множина обов'язково повинна містити 0. Для того щоб $N5$ перетворити в групу $GN5$, необхідно коректно задати значення для операції додавання до одного з елементів групи, скажімо до 1. Дійсно, оскільки $a + 0 = a$ для всякого a із $GN5$, то перший рядок таблиці додавання елементів групи визначений (табл.1), а внаслідок симетричності (оскільки $GN5$ абелева) — і перший стовпчик цієї таблиці. Нехай, наприклад, задано $0 + 1 = 1, 1 + 1 = 4, 1 + 4 = 2, 1 + 2 = 3, 1 + 3 = 0$. Таке задання коректне, оскільки маємо єдиність результату (але єдиність результату, як буде показано нижче, не достатня умова гарантії коректності). Тепер послідовно знаходимо:

$$\begin{aligned} 2 + 2 &= 2 + (1 + 4) = (2 + 1) + 4 = \\ &= 3 + (1 + 1) = (3 + 1) + 1 = 0 + 1 = 1, \\ 2 + 3 &= 2 + (1 + 2) = (2 + 2) + 1 = 1 + 1 = 4, \\ 2 + 4 &= 2 + (1 + 1) = (2 + 1) + 1 = 3 + 1 = 0, \\ 3 + 3 &= 3 + (1 + 2) = (3 + 1) + 2 = 0 + 2 = 2, \\ 3 + 4 &= 3 + (1 + 1) = (3 + 1) + 1 = 0 + 1 = 1, \\ 4 + 4 &= 4 + (1 + 1) = (4 + 1) + 1 = 2 + 1 = 3. \end{aligned}$$

Заносимо ці значення в табл. 2 і на цьому закінчимо побудову групи $GN5$.

Таблиця 1

+	0	1	2	3	4
0	0	1	2	3	4
1	1	4	3	0	2
2	2	3			
3	3	0			
4	4	2			

Таблиця 2

+	0	1	2	3	4
0	0	1	2	3	4
1	1	4	3	0	2
2	2	3	1	4	0
3	3	0	4	2	1
4	4	2	0	1	3

Аналогічно можна задати і всяку іншу групу $GN5$, що містить 0. Дійсно, якщо, наприклад, $M5 = \{0, 2, 6, 3, 5\}$, то встановимо взаємно однозначну відповідність h між $N5$ і $M5$ (ция відповідність існує, оскільки $N5$ і $M5$ рівнопотужні), яка переводить 0 в 0, а решту елементів довільним чином. Наприклад, нехай задано $1 \Leftrightarrow 6, 2 \Leftrightarrow 3, 3 \Leftrightarrow 5, 4 \Leftrightarrow 2, 0 \Leftrightarrow 0$. Звідси отримуємо табл. 3.

Перевіримо, наприклад, чому відповідає в цій таблиці $4 = 1 + 1$. Маємо $h(1 + 1) = h(4) = h(1) + h(1) = 6 + 6 = 2$.

Як вправа, пропонується виконати перевірку всієї табл. 3.

Таблиця 3

+	0	2	6	3	5
0	0	2	6	3	5
2	2	5	3	0	6
6	6	3	2	5	0
3	3	0	5	6	2
5	5	6	0	2	3

Зауважимо, що для того щоб побудувати групу GNk , мало вимагати тільки однозначності операції додавання. Якщо задати додавання в групі так, що в ній буде елемент скінченного порядку (тобто група з крученнем), то коректність визначення операції може бути порушена. Наприклад, якщо задати додавання

$$0 + 1 = 1, 1 + 1 = 0, 1 + 2 = 3, 1 + 3 = 4, 1 + 4 = 2,$$

то, виконуючи перевірку (обчислюючи) $1 + 3$, отримаємо

$$1 + 3 = 1 + (1 + 2) = (1 + 1) + 2 = 0 + 2 = 2,$$

що не збігається з визначенням вище. Справа в тому, що елемент 1 є елементом порядку 2 ($1 + 1 = 0$), і це вносить свої корективи при визначенні операції (її вже не можна задавати довільно). ▲

7. ВІЛЬНЕ КІЛЬЦЕ

Вільна група $T(\Omega, X, Eq)$ називається *вільним кільцем*, якщо Eq визначає $T(\Omega, X, Eq)$ як:

- 1) вільну абелеву групу відносно додавання;
- 2) вільний групоїд відносно множення;
- 3) таку, в якій виконуються закони дистрибутивності, тобто для будь-яких x, x', x'' із $T(\Omega, X, Eq)$

$$\begin{aligned} x(x' + x'') &= (xx') + (xx''), \\ (x + x')x'' &= (xx'') + (x'x''). \end{aligned}$$

Інакше кажучи, Ω — це чотири операції: бінарні операції додавання і множення, унарну операцію взяття оберненого відносно операції додавання і нульарну операцію, яка фіксує нульовий елемент абелевої групи кільця. Цей нульовий елемент називається *нульовим* елементом кільця.

Множення у вільному кільці зводиться, згідно з законами дистрибутивності, до множення елементів із X , яке виконується за правилами множення слів у вільному групоїді.

Із 1—3 випливають співвідношення, які дає таке твердження.

Твердження 2.2.4. У довільному кільці K для всяких його елементів a, b, c справедливі співвідношення:

- 1) $a(b - c) = ab - ac, (b - c)a = ba - ca;$
- 2) $a0 = 0a = 0;$
- 3) $(-a)b = a(-b) = -ab, (-a)(-b) = ab.$

Д о в е д е н н я. 1. Згідно із законами комутативності і асоціативності операції додавання можна записати

$$c + (b - c) = b.$$

Помноживши обидві частини цього рівняння ліворуч на a , маємо

$$a(c + (b - c)) = ac + a(b - c) = ab.$$

Звідси $(ac + a(b - c)) - ac = ab - ac = a(b - c)$, знову ж таки згідно із законами комутативності і асоціативності операції додавання.

Аналогічно доводиться і друге співвідношення.

2. Дійсно, нехай b — довільний елемент кільця. Тоді, згідно з твердженням 1, доведеним вище, маємо

$$a0 = a(b - b) = ab - ab = 0.$$

3. Доведення пропонується як вправа.

Зауважимо, що обернене твердження до твердження 2 в довільному кільці хибне, тобто існують такі кільця, в яких є відмінні від нуля елементи a, b , добуток яких дорівнює нулю ($ab = 0$). Якщо такі елементи в кільці є, то вони називаються *дільниками нуля*.

Кільце називається *асоціативним (комутативним)*, якщо операція множення асоціативна (комутативна), і *кільцем з одиницею*, коли має одиничний елемент відносно операції множення.

Кільце називається *асоціативно-комутативним*, якщо воно асоціативне і комутативне одночасно. Елементами вільного асоціативно-комутативного кільця з множиною вільних твірних X є многочлени від елементів із X з цілими коефіцієнтами. Тому таке кільце часто називають просто *кільцем многочленів* над X .

Множення у вільному асоціативному кільці виконується за правилом множення слів у вільній напівгрупі, а сама напівгрупа називається *мультиплікативною напівгрупою* асоціативного кільця.

Приклади 2.2.7 (кілек)

1. Прикладом асоціативного кільця з одиницею може служити множина квадратних матриць над довільним кільцем P з одиницею.

Нехай $M(p, q, P)$ — множина всіх матриць розмірності $p \times q$ над кільцем P , а $M(p, P)$ — множина квадратних матриць $M(p, p, P)$.

Матриця C , яка складається з елементів $a_{ij} + b_{ij}$, де a_{ij}, b_{ij} — відповідно елементи матриць A і B із $M(p, q, P)$, називається **сумою матриць A і B** . Оскільки множина P є кільцем, то множина $M(p, q, P)$, очевидно, є абелевою групою відносно додавання. Роль нульового елемента відіграє матриця, в якої всі елементи — нулі (**нульова матриця**), а роль оберненої матриці для матриці A відіграє матриця A' , в якій $a'_{ij} = -a_{ij}$, де a'_{ij}, a_{ij} — елементи матриць A' і A відповідно.

Нехай $A \in M(p, q, P)$, $B \in M(q, r, P)$. Тоді матриця C , яка складається з елементів

$$c_{ij} = \sum_{k=1}^q a_{ik} \cdot b_{kj},$$

називається **добутком матриць A і B** . З цього означення випливає, що не всякі дві матриці можна перемножити, а лише відповідні, тобто такі, в яких число стовпчиків першої дорівнює числу рядків другої. Ясно, що квадратні матриці з $M(p, P)$ відповідні, і для таких матриць добуток завжди визначений.

Розглянемо множину матриць $M(p, P)$. Матриця A називається **одиничною**, якщо $a_{ij} = 0$ при $i \neq j$ і $a_{ii} = 1_p$, де 1_p — одиниця кільця P . Одинична матриця, як правило, позначається E . Легко впевнитися, що $\forall A \in M(p, P)$ виконується рівність $A \cdot E = E \cdot A$, тобто матриця E відіграє роль одиничного елемента в множині $M(p, P)$.

Неважко показати, що $M(p, P)$ є асоціативним кільцем з одиницею. Покажемо, наприклад, справедливість одного з дистрибутивних законів:

$$(\forall A, B \in M(p, P)) A \cdot (B + C) = A \cdot B + A \cdot C.$$

Дійсно, за означенням добутку матриць маємо

$$\sum_{k=1}^p a_{ik} (b_{kj} + c_{kj}) = \sum_{k=1}^p (a_{ik} b_{kj} + a_{ik} c_{kj}) = \sum_{k=1}^p a_{ik} b_{kj} + \sum_{k=1}^p a_{ik} c_{kj},$$

тобто $A \cdot (B + C)$ і $A \cdot B + A \cdot C$ мають однакові елементи.

2. Множина $M = \{0, 1\}$ буде кільцем, якщо визначити додавання і множення елементів цієї множини так:

$$0 + 0 = 1 + 1 = 0, \quad 0 + 1 = 1 + 0 = 1, \quad 0 \cdot 1 = 1 \cdot 0 = 0, \quad 1 \cdot 1 = 1.$$

Неважко перевірити, що дана множина буде асоціативно-комутативним кільцем з одиницею.

Розглядаючи квадратну матрицю над цим кільцем, яка відповідає деякому бінарному відношенню R на скінченній множині A

(див. розділ 1), можна обчислити транзитивне замикання цього відношення, користуючись його матрицею $A(R)$.

Розглянемо приклад. Нехай $A = \{a_1, a_2, a_3, a_4\}$, $R = \{(a_1, a_1), (a_1, a_2), (a_1, a_3), (a_2, a_4), (a_3, a_4), (a_4, a_1), (a_4, a_3)\}$. Тоді матрицю $A(R)$ запишемо так:

$$A(R) = \begin{vmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{vmatrix}.$$

Пропонується показати, що $A(R)^2$ відповідає відношенню R^2 , $A(R)^3$ — відношенню R^3 і т.д.

Загалом, якщо R і R_1 — деякі бінарні відношення на множині A , яким відповідають матриці $A(R)$ і $A(R_1)$, то відношенню $R * R_1$ відповідає матриця, яка є добутком матриць $A(R)$ і $A(R_1)$ (див. вправу 13).

3. При розгляді абелевих груп було побудовано приклад *нетрадиційної арифметики* для операції додавання (див. приклад 2.2.6). Кільця дозволяють розширити таку побудову і на операцію множення.

Розглянемо спочатку, як довизначається група $GN5$ (див. табл. 2) до кільця з одиницею. Роль одиниці буде відігравати 1. Згідно з аксіомами кільця і твердженням 2.2.4 маємо: для всякого елемента a із $GN5$ $a \cdot 1 = 1 \cdot a = a$, $0 \cdot a = a \cdot 0 = 0$. Таким чином, два рядки і два стовпчики таблиці множення вже визначені. Далі, використовуючи табл. 2 для додавання і закон дистрибутивності, одержуємо

$$\begin{aligned} 4 \cdot 2 &= (1 + 1) \cdot 2 = 2 + 2 = 1; \\ 4 \cdot 3 &= (1 + 1) \cdot 3 = 3 + 3 = 2; \\ 4 \cdot 4 &= (1 + 1) \cdot 4 = 4 + 4 = 3. \end{aligned}$$

Легко бачити, що
 $4 \cdot 2 = 2 \cdot 4$ і $4 \cdot 3 = 3 \cdot 4$,
тобто елемент 4 комутативний з рештою елементів.

Далі $2 \cdot 2 = (1 + 4) \cdot 2 = 2 + 4 \cdot 2 = 2 + 1 = 3$;
 $2 \cdot 3 = (1 + 4) \cdot 3 = 3 + 4 \cdot 3 = 3 + 2 = 4$;
 $2 \cdot 4 = (1 + 4) \cdot 4 = 4 + 4 \cdot 4 = 4 + 3 = 1$.

Аналогічно одержуємо $3 \cdot 3 = (2 + 1) \cdot 3 = 2 \cdot 3 + 3 = 4 + 3 = 1$ і решту елементів табл. 4. Із симетричності таблиці випливає, що множина елементів $GN5$ комутативна. Крім того, легко перевіри-

Таблиця 4

.	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	3	4	1
3	0	3	4	1	2
4	0	4	1	2	3

ти, що GN_5 також і асоціативна, тобто GN_5 — асоціативно-комутативне кільце з одиницею.

Розглянемо тепер приклад кільця без одиниці. Візьмемо групу GMS , операція додавання якої задана табл. 3. Для того щоб перетворити цю групу в кільце, необхідно, як і у випадку додавання, задати хоча б один рядок для операції множення.

Нехай задано: $2 \cdot 2 = 3$, $2 \cdot 3 = 5$, $2 \cdot 5 = 6$, $2 \cdot 6 = 2$. Тоді

Таблиця 5

	0	2	6	3	5
0	0	0	0	0	0
2	0	3	2	5	6
6	0	5	6	2	3
3	0	2	3	6	5
5	0	6	5	3	2

маємо:

$$\begin{aligned} 5 \cdot 2 &= (2 + 2) \cdot 2 = \\ &= 2 \cdot 2 + 2 \cdot 2 = 3 + 3 = 6; \\ 5 \cdot 3 &= (2 + 2) \cdot 3 = \\ &= 2 \cdot 3 + 2 \cdot 3 = 5 + 5 = 3; \\ 5 \cdot 6 &= (2 + 2) \cdot 6 = \\ &= 2 \cdot 6 + 2 \cdot 6 = 2 + 2 = 5; \end{aligned}$$

$$\begin{aligned} 5 \cdot 5 &= (2 + 2) \cdot 5 = 2 \cdot 5 + 2 \cdot 5 = 6 + 6 = 2; \\ 6 \cdot 2 &= (5 + 2) \cdot 2 = 5 \cdot 2 + 2 \cdot 2 = 6 + 3 = 5; \\ 6 \cdot 3 &= (5 + 2) \cdot 3 = 5 \cdot 3 + 2 \cdot 3 = 3 + 5 = 2; \\ 6 \cdot 5 &= (5 + 2) \cdot 5 = 5 \cdot 5 + 2 \cdot 5 = 2 + 6 = 3; \\ 6 \cdot 6 &= (5 + 2) \cdot 6 = 5 \cdot 6 + 2 \cdot 6 = 5 + 2 = 6; \\ 3 \cdot 2 &= (6 + 2) \cdot 2 = 6 \cdot 2 + 2 \cdot 2 = 5 + 3 = 2; \\ 3 \cdot 3 &= (6 + 2) \cdot 3 = 6 \cdot 3 + 2 \cdot 3 = 2 + 5 = 6; \\ 3 \cdot 6 &= (6 + 2) \cdot 6 = 6 \cdot 6 + 2 \cdot 6 = 6 + 2 = 3; \\ 3 \cdot 5 &= (6 + 2) \cdot 5 = 6 \cdot 5 + 2 \cdot 5 = 3 + 6 = 5. \end{aligned}$$

Остаточно операція множення представлена табл. 5. З таблиці видно, що кільце некомутативне. Легко впевнитися, що воно і неасоціативне. Це видно з такого прикладу: результати $(3 \cdot 6) \cdot 5 = 3 \cdot 5 = 5$ і $3 \cdot (6 \cdot 5) = 3 \cdot 3 = 6$ не рівні між собою.

Зауважимо, що для того щоб оптимально знаходити таблицю як для додавання, так і для множення, найкраще зробити таким чином. Якщо визначено множення на число 2 (як в попередньому прикладі), то в таблиці додавання знаходимо суму $2 + 2$ (вона дорівнює 5) і беремо результат цієї суми як перший множник. Далі робимо аналогічно для числа 2 і числа, яке отримане раніше (в прикладі це число 5), шукаємо добуток цих чисел. Знаходимо результат множення для них і т.д., доки не побудуємо всі елементи таблиці.

При обчисленнях на ЕОМ така методика часто відіграє дуже важливу роль для побудови ефективних алгоритмів.

4. Асоціативно-комутативне кільце називається **булевим**, якщо для всякого елемента a із цього кільця справедливе співвідношення $a^2 = a$. Ця тотожність називається **законом ідемпотентності**.

Твердження 2.2.5. У всякому булевому кільці для будь-яких елементів a, b виконуються співвідношення:

$$(a) 2 \cdot a = 0; \quad (b) a \cdot b = b \cdot a.$$

Д о в е д е н н я. Доведемо (а). Внаслідок ідемпотентності $a + b = (a + b)^2 = a + a * b + b * a + b$. Звідси отримуємо $a * b + b * a = 0$. Покладемо $b = a$, тоді $a^2 + a^2 = a + a = 2 * a = 0$. Звідси знаходимо також, що $a = -a$.

Доведемо (б). Із (а) одержуємо, що $a * b = b * a = 0$. Міняючи a на $-a$ в одному з доданків, знаходимо $a * b - b * a = 0$ або $a * b = b * a$, що і потрібно було довести. \blacktriangleleft

Асоціативно-комутативне кільце без дільників нуля називається **областю цілісності**.

Областями цілісності, як легко переконатися, є кільця цілих і раціональних чисел — \mathbf{Z} і \mathbf{RC} . Областю цілісності буде також і вільне кільце многочленів.

Останній приклад з многочленами підказує конструкцію, за допомогою якої можна будувати інші приклади областей цілісності.

Нехай KC — довільне асоціативно-комутативне кільце. Розглянемо всі можливі многочлени

$$a_0 + a_1 * x + a_2 * x^2 + \dots + a_n * x^n,$$

відносно невідомого x з коефіцієнтами a_i з \mathbf{RC} , де $a_i \in \mathbf{N}$, $i = 1, 2, \dots, n$. Якщо $a_n \neq 0$, то число n називається **степенем** цього многочлена. Визначаючи додавання і множення многочленів звичайним шляхом, так, як це прийнято в курсі вищої алгебри, отримуємо **кільце многочленів** $R[x]$ від параметра x над кільцем KC . Нулем кільця $R[x]$ служить многочлен, всі коефіцієнти якого дорівнюють нулю.

Аналогічно можна визначити кільце многочленів $R[x_1, x_2, \dots, x_n]$ від будь-якого скінченного числа невідомих, як кільце многочленів від одного невідомого x_n над кільцем $R[x_1, x_2, \dots, x_{n-1}]$.

Справедлива така теорема.

Теорема 2.2.3. Якщо KC — область цілісності, то $R[x_1, x_2, \dots, x_n]$ — теж область цілісності (див. вправу 17).

Асоціативно-комутативне кільце з одиницею називається **полям**, якщо воно як відносно додавання, так і відносно множення є абелевою групою. Група поля відносно додавання називається **адитивною**, а група поля відносно множення — **мультиплікативною**.

Зауважимо, що поле є обlastю цілісності, і через це, як випливає з теореми 2.2.3, кільце многочленів над довільним полем P є обlastю цілісності. Одиницею в цій обlastі служить многоч-

лен, коефіцієнти якого дорівнюють нулю, крім коефіцієнта a_0 , який дорівнює 1_p — одиниці поля P .

8. ВЕКТОРНІ ПРОСТОРИ

Вільна група $T(\Omega, X, Eq)$ є векторним простором над деяким полем P , якщо Ω містить бінарну операцію додавання елементів (2.2.3), нескінченне число унарних операцій множення елементів (2.2.3) на елементи з поля P (для кожного a з P своя операція) і нульарну операцію, яка фіксує нульовий елемент. Множина Eq — це всі співвідношення, що визначають $T(\Omega, X, Eq)$ як вільну абелеву групу, тобто для довільних x, x', x'' із $T(\Omega, X, Eq)$ маємо

$$\begin{aligned} x + (x' + x'') &= (x + x') + x'', \quad x + 0 = x, \\ x + x' &= x' + x, \quad x + (-x) = 0, \end{aligned}$$

а також для довільних x, x' із $T(\Omega, X, Eq)$ і a, b із P виконуються рівності

$$\begin{aligned} a \cdot (x + x') &= a \cdot x + a \cdot x'; \\ (a + b) \cdot x &= a \cdot x + b \cdot x; \\ a \cdot (b \cdot x) &= (a \cdot b) \cdot x. \end{aligned}$$

Елементами $T(\Omega, X, Eq)$ є елементи, що мають вигляд (2.2.3) і називаються **векторами**. Символам x_i із X ставиться у відповідність вектор

$$0 \cdot x_1 + 0 \cdot x_2 + \dots + 1 \cdot x_i + \dots + 0 \cdot x_n + \dots, \quad (2.2.4)$$

а роль нульового вектора простору відіграє елемент

$$0 \cdot x_1 + 0 \cdot x_2 + \dots + 0 \cdot x_i + \dots + 0 \cdot x_n + \dots = 0,$$

де $i = 1, 2, \dots$

Нехай $v_1, v_2, \dots, v_n \in T(\Omega, X, Eq)$ і $m_1, m_2, \dots, m_n \in P$. Тоді суму

$$m_1 \cdot v_1 + m_2 v_2 + \dots + m_n v_n = \sum_{i=1}^n m_i \cdot v_i$$

називають **лінійною комбінацією векторів** v_1, v_2, \dots, v_n . Говорять, що вектори v_1, v_2, \dots, v_n лінійно незалежні, якщо

$$\sum_{i=1}^n m_i \cdot v_i = 0 \Leftrightarrow m_1 = m_2 = \dots = m_n = 0,$$

у протилежному випадку вектори v_1, v_2, \dots, v_n називаються **лінійно залежними**.

Припустимо, що в деякому просторі L над полем P існує лінійно незалежна система векторів v_1, v_2, \dots, v_n , і немає жодної

лінійно незалежної системи, яка складається з більшого, ніж n числа векторів. Тоді говорять, що L — n -вимірний векторний простір над полем P . Число n називають розмірністю простору L , а довільну сукупність лінійно незалежних векторів v_1, v_2, \dots, v_n із L — базисом L . Якщо такого числа n немає, то простір L називається **нескінченновимірним**.

Зауважимо, що вектори (2.2.4) лінійно незалежні, і тому X можна розглядати як базис $T(\Omega, X, Eq)$, і якщо X — скінченна множина, то $T(\Omega, X, Eq)$ — скінченновимірний векторний простір над P .

Теорема 2.2.4. Якщо X — базис n -вимірного векторного простору $T(\Omega, X, Eq)$ над полем P , то для всякого вектора $u \in T(\Omega, X, Eq)$ існує єдина можливість запису його у вигляді лінійної комбінації $u = m_1 \cdot x_1 + m_2 \cdot x_2 + \dots + m_n \cdot x_n$ базисних векторів $x_1, x_2, \dots, x_n \in X$.

Д о в е д е н н я. Оскільки X — базис, то сукупність векторів u, x_1, x_2, \dots, x_n в n -вимірному векторному просторі $T(\Omega, X, Eq)$ лінійно залежна, в той час як x_1, x_2, \dots, x_n — лінійно незалежна система.

Отже, $u = m_1 \cdot x_1 + m_2 \cdot x_2 + \dots + m_n \cdot x_n$.

Коли б існувала інша можливість запису вектора u , наприклад $u = l_1 \cdot x_1 + l_2 \cdot x_2 + \dots + l_n \cdot x_n$, то ми мали б

$$(m_1 - l_1) \cdot x_1 + (m_2 - l_2) \cdot x_2 + \dots + (m_n - l_n) \cdot x_n = 0.$$

Звідси, оскільки вектори x_1, x_2, \dots, x_n лінійно незалежні, одержуємо $m_1 = l_1, m_2 = l_2, \dots, m_n = l_n$.

Теорема доведена.

Однозначно визначені коефіцієнти m_1, m_2, \dots, m_n вектора u з $T(\Omega, X, Eq)$ називаються його **координатами** в базисі v_1, \dots, v_n .

Якщо $L \subseteq T(\Omega, X, Eq)$, то множина L називається **підпростором** простору $T(\Omega, X, Eq)$, коли вона разом з векторами v_1, v_2, \dots, v_k містить і всяку їх лінійну комбінацію $m_1 \cdot v_1 + m_2 \cdot v_2 + \dots + m_k \cdot v_k$. В окремому випадку, якщо $T(\Omega, X, Eq)$ — n -вимірний векторний простір і L — його підпростір, то ясно, що розмірність L не перевищує розмірності всього простору $T(\Omega, X, Eq)$. Справедлива така теорема.

Теорема 2.2.5. Якщо підпростір L n -вимірного векторного простору $T(\Omega, X, Eq)$ має ту ж розмірність, що і $T(\Omega, X, Eq)$, то він збігається з усім простором $T(\Omega, X, Eq)$.

Д о в е д е н н я. Нехай L і $T(\Omega, X, Eq)$ мають одну і ту ж розмірність n . Візьмемо базис простору L — v_1, v_2, \dots, v_n . Ця система буде базисом обох просторів (оскільки $v_1, v_2, \dots, v_n \in T(\Omega, X, Eq)$). Отже, для всякого $u \in T(\Omega, X, Eq)$ існує розклад

$$u = m_1 \cdot v_1 + m_2 \cdot v_2 + \dots + m_n \cdot v_n$$

і $u \in L$, оскільки u — лінійна комбінація векторів із L , тобто $T(\Omega, X, Eq) \subseteq L$, а значить, $L = T(\Omega, X, Eq)$.

9. БУЛЕВІ АЛГЕБРИ

Алгебра $G = (A, \Omega) \in K(\Omega, Eq)$ називається *булевою алгеброю*, якщо Ω складається з:

- 61) двох бінарних операцій — \vee (або) і $\&$ (і);
- 62) однієї унарної операції — \neg (заперечення);
- 63) двох нульарних операцій — 0 (нуль) і 1 (одиниця),

і для довільних $a, b, c \in A$ виконується така сукупність співвідношень Eq :

- c1) $a \vee b = b \vee a, a \& b = b \& a$ — комутативність;
- c2) $a \vee (b \vee c) = (a \vee b) \vee c,$
- $a \& (b \& c) = (a \& b) \& c$ — асоціативність;
- c3) $a \vee (b \& c) = (a \vee b) \& (a \vee c),$
- $a \& (b \vee c) = (a \& b) \vee (a \& c)$ — дистрибутивність;
- c4) $a \vee 0 = a, a \vee \neg a = 1, a \& 1 = a, a \& \neg a = 0$ — закони для нуля, одиниці і заперечення.

Як наслідки співвідношень c1—c4, можна одержати деякі важливі співвідношення.

Наслідок 2.2.1. (Закон ідемпотентності.) Для всякого елемента a з A виконуються рівності

$$a \vee a = a \& a = a, \quad a \vee 1 = 1, \quad a \& 0 = 0.$$

Д о в е д е н н я. Використовуючи закони c4 і закони дистрибутивності, маємо

$$\begin{aligned} a \vee a &= (a \vee a) \& 1 = (a \vee a) \& (a \vee \neg a) = \\ &= a \vee (a \& \neg a) = a \vee 0 = a, \\ a \& a &= (a \& a) \vee 0 = (a \& a) \vee (a \& \neg a) = \\ &= a \& (a \vee \neg a) = a \& 1 = a. \end{aligned}$$

Звідси, а також із законів асоціативності випливає

$$\begin{aligned} a \vee 1 &= a \vee (a \vee \neg a) = (a \vee a) \vee \neg a = a \vee \neg a = 1; \\ a \& 0 &= a \& (a \& \neg a) = (a \& a) \& \neg a = a \& \neg a = 0. \end{aligned}$$

Наслідок 2.2.2. (Закон поглинання.) Для будь-яких елементів $a, b \in A$ виконуються рівності

$$a \& (a \vee b) = a \vee (a \& b) = a.$$

Д о в е д е н н я. Доведемо рівність $a \vee (a \& b) = a$. Використовуючи співвідношення c4, закон дистрибутивності і наслідок 2.2.1, маємо

$$a \vee (a \& b) = (a \& 1) \vee (a \& b) = a \& (1 \vee b) = a \& 1 = a.$$

Аналогічно доводиться і друга рівність:

$$a \& (a \vee b) = (a \vee 0) \& (a \vee b) = a \vee (0 \& b) = a \vee 0 = a.$$

Властивість 2.2.1. Для довільних елементів $a, b, c \in A$ із рівностей $a \& c = b \& c$ і $a \vee c = b \vee c$ випливає рівність $a = b$.

Доведення. За законом поглинання маємо $a = a \vee (a \& c)$. Застосовуючи закони поглинання, дистрибутивності і рівності, які наведені в умові, отримуємо

$$\begin{aligned} a &= a \vee (a \& c) = a \vee (b \& c) = (a \vee b) \& (a \vee c) = \\ &= (a \vee b) \& (b \vee c) = b \vee (a \& c) = b \vee (b \& c) = b, \end{aligned}$$

що і було потрібно довести.

Наслідок 2.2.3. (Закон подвійного заперечення.) Для довільного елемента a із A виконуються рівності

$$\neg(\neg a) = a, \neg(0) = 1, \neg(1) = 0.$$

Дійсно, $\neg(a) \vee \neg(\neg(a)) = 1$ і $\neg(a) \& \neg(\neg(a)) = 0$, $\neg(a) \vee a = 1$ і $\neg(a) \& a = 0$ внаслідок с4. Звідси випливає, що $\neg(a) \vee \neg(\neg(a)) = \neg(a) \vee a$ і $\neg(a) \& \neg(\neg(a)) = \neg(a) \& a$. За властивістю 2.2.1 отримуємо $a = \neg(\neg(a))$.

Аналогічно із $\neg 0 \vee 0 = 1 \vee 0 = 1$ і $\neg 0 \& 0 = 1 \& 0 = 0$ випливає $\neg 0 = 1$.

Наслідок 2.2.4. (Закони де Моргана.) Для довільних елементів $a, b \in A$ виконуються рівності

$$\begin{aligned} \neg(a \vee b) &= \neg a \& \neg b, \\ \neg(a \& b) &= \neg a \vee \neg b. \end{aligned}$$

Доведення. Розглянемо вираз

$$\begin{aligned} (a \vee b) \& (\neg a \& \neg b) &= (a \& (\neg a \& \neg b)) \vee (b \& (\neg a \& \neg b)) = \\ &= 0 \vee 0 = 0. \end{aligned}$$

Далі

$$\begin{aligned} (a \vee b) \vee (\neg a \& \neg b) &= ((a \vee b) \vee \neg a) \& ((a \vee b) \vee \neg b) = \\ &= 1 \& 1 = 1. \end{aligned}$$

Згідно з с4 маємо

$$(a \vee b) \& (\neg a \& \neg b) = (a \vee b) \& \neg(a \vee b) = 0$$

i

$$(a \vee b) \vee (\neg a \& \neg b) = (a \vee b) \vee \neg(a \vee b) = 1.$$

Звідси за властивістю 2.2.1 робимо висновок, що $\neg(a \vee b) = \neg a \& \neg b$.

Аналогічно доводиться і другий закон.

Операції булевої алгебри \vee , $\&$, \neg називають відповідно **диз'юнкцією**, **кон'юнкцією** і **запереченнем**, а також **пропозиційними (булевими) зв'язками**.

Приклади 2.2.8 (булевих алгебр)

1. Булеан множини. Прикладом булевої алгебри може служити множина $B(U)$ всіх підмножин деякої множини U зі звичайними операціями об'єднання, перетину і доповнення в множині $B(U)$. Роль нуля в цьому випадку відіграє пуста множина, а одиниці — вся множина U . Дійсно, перелічені операції над множинами відповідають $M1$ — $M5$, які є законами булевої алгебри.

2. Алгебра булевих функцій. Алфавіт $X = \{x_1, x_2, \dots, x_n\}$ називатимемо **алфавітом булевих змінних**, якщо кожна змінна $x_i \in X$ ($i = 1, 2, \dots, n$) може набувати лише одне з двох значень: 0 або 1, тобто область інтерпретації змінних із множини X є множина $\{0, 1\}$. Константи 0 і 1 називають у цьому випадку булевими константами.

m-Арна функція $f: X^m \rightarrow \{0, 1\}$ називається **булевою функцією над алфавітом X** , де $X = \{x_1, x_2, \dots, x_n\}$ — алфавіт булевих змінних. З означення булевої функції від m аргументів випливає, що її область визначення скінчена і складається рівно з 2^m елементів. Довільний такий елемент (який називають набором) — це вектор довжини m , кожна компонента якого дорівнює 0 або 1. Виходячи з того, що всяка булева функція теж приймає лише одне з двох значень (0 або 1), можна підрахувати число всіх різних m -арних булевих функцій. Дійсно, оскільки на кожному наборі з області визначення m -арної функції вона може приймати значення 0 або 1 незалежно від значень на інших наборах, то, поступово поширюючи область визначення з одного набору на всі 2^m наборів, ми щоразу подвоюємо число різних функцій. Таким чином, існує 2^{2^m} різних булевих функцій від m змінних, зокрема, 4 булевих функції однієї змінної, 16 булевих функцій двох змінних і т.д. Завдяки тому, що області зміни аргументів і області значень булевих функцій збігаються, можна будувати суперпозиції булевих функцій, виконуючи підстановки одних із них на місце інших. У зв'язку з цим всяка булева функція f може розглядатися як операція над множиною всіх булевих функцій. Виправивши деяку множину функцій за множину основних операцій, можна будувати різні алгебри булевих функцій. При цьому вирази в такій алгебрі будуть представляти відповідні булеві функції. Наприклад, вираз $S(x_1, x_2, \dots, x_k)$ представляє функцію $f(x_1, x_2, \dots, x_k)$, якщо на всікому наборі значень змінних x_1, x_2, \dots, x_k значення S і f збігаються. Особливе місце, як ми переконаємо

пізніше, займають вже відомі нам булеві операції (функції), що мають вигляд

$$f(x) = \neg x, g(x, y) = x \vee y, h(x, y) = x \wedge y.$$

Набір операцій над заданим алфавітом булевих змінних $X = \{x_1, x_2, \dots, x_n\}$ називають **функціонально повним**, якщо всяка булева функція над алфавітом X може бути представлена хоча б одним виразом в алгебрі, яка визначається цим набором операцій. Закони булевої алгебри показують, що булева функція може мати кілька виразів, які її представляють. У зв'язку з цим виникає питання про канонічну форму виразів булевої алгебри. Розглянемо дві такі форми — кон'юнктивну і диз'юнктивну. Наведемо необхідні означення.

Нехай $X = \{x_1, x_2, \dots, x_n\}$ — алфавіт булевих змінних, а фіксованими булевими операціями є операції диз'юнкції, кон'юнкції і заперечення.

Елементарним добутком називається вираз, який являє собою кон'юнкцію довільного скінченного числа попарно різних символів булевих змінних із X , частина яких (можливо, пуста) знаходиться під знаком заперечення. Наприклад, $\neg x \wedge y \wedge z$, $\neg x$, $x \wedge \neg y$, де $x, y, z \in X$, є елементарними добутками. До елементарних добутків також відносять і константу 1, про яку говорять, що вона складається з пустої множини членів.

Диз'юнктивною нормальню формою (ДНФ) називається диз'юнкція довільної скінченної множини попарно різних елементарних добутків. Це означення включає в себе як випадок пустої множини членів, так і випадок одного члена — булевої константи. Основну властивість ДНФ дає таке твердження.

Теорема 2.2.6. *Всякий вираз булевої алгебри можна перетворити в еквівалентну йому диз'юнктивну нормальну форму.*

Доведення теореми, по суті, являє собою алгоритм побудови ДНФ для заданого виразу. Опишемо його у вигляді кроків, які необхідно виконати в процесі побудови ДНФ. Кожний із цих кроків досить зрозумілий, і тому ми обмежимося лише їх послідовністю, перший крок якої застосовується до початкового виразу, а кожний наступний — до результату попереднього.

1. За допомогою законів де Моргана вираз перетворюється у вираз, в якому немає символу заперечення ні перед диз'юнкцією, ні перед кон'юнкцією підвиразів даного виразу.

2. Застосовуємо закон подвійного заперечення.

3. Застосовуємо закони дистрибутивності і закон комутативності для кон'юнкції доти, доки вираз не матиме вигляду диз'юнкції кількох добутків.

4. Приводимо кожний диз'юнктивний член або до елементарного добутку, або до константи.

5. Застосовуючи закони для нуля і одиниці, спрошуємо одержаний вираз. Це і є шукана ДНФ.

Теорема доведена.

Приклад 2.2.9

Побудувати ДНФ для виразу $\neg(x \wedge \neg y) \wedge ((x \wedge z) \vee \neg(y \vee z))$. Застосовуючи двічі закони де Моргана, даний вираз можна переворити до такого виразу:

$$\neg(x \wedge \neg y) \wedge ((x \wedge z) \vee \neg(y \vee z)) = (\neg x \vee \neg \neg y) \wedge ((x \wedge z) \vee (\neg y \wedge \neg z)).$$

Застосовуючи закон подвійного заперечення, одержуємо таку формулу:

$$(\neg x \vee y) \wedge ((x \wedge z) \vee (\neg y \wedge \neg z)).$$

Розкриваючи дужки за допомогою законів дистрибутивності, маємо

$$\begin{aligned} & (\neg x \vee y) \wedge ((x \wedge z) \vee (\neg y \wedge \neg z)) = \\ & = (\neg x \vee y) \wedge (x \wedge z) \vee ((\neg x \vee y) \wedge (\neg y \wedge \neg z)) = \\ & = (x \wedge z) \wedge (\neg x \vee y) \vee (\neg y \wedge \neg z) \wedge (\neg x \vee y) = \\ & = (x \wedge z \wedge \neg x) \vee (x \wedge y \wedge z) \vee (\neg x \wedge \neg y \wedge \neg z) \vee (y \wedge \neg y \wedge \neg z). \end{aligned}$$

Нарешті, скориставшись законами $x \wedge \neg x = 0$, $0 \wedge x = 0$ і $0 \vee x = x$, остаточно знаходимо, що

$$\neg(x \wedge \neg y) \wedge ((x \wedge z) \vee \neg(y \vee z)) = (x \wedge y \wedge z) \vee (\neg x \wedge \neg y \wedge \neg z).$$

Виникає питання: чи буде ДНФ для заданої булевої функції єдиним представленням, чи ні? Відповідь у загальному випадку негативна. Дійсно, наприклад, ліві і праві частини закону поглинання задовольняють означення ДНФ, але відрізняються одною від одної. Можна ввести умови, при виконанні яких диз'юнктивна нормальна форма булевої функції буде єдиною. Однією з таких форм є досконала ДНФ.

Елементарний добуток називається *конституентою одиниці* для алфавіту булевих змінних X , якщо він містить (під знаком заперечення чи без нього) всі змінні з X . Ясно, що кожна конституента одиниці приймає значення 1 лише на єдиному наборі з 2^m наборів для булевих змінних.

ДНФ булевої функції $f(x_1, x_2, \dots, x_n)$ називається *досконалою* (ДДНФ), якщо всі її елементарні добутки — конституенти одиниці для множини $\{x_1, x_2, \dots, x_n\}$ аргументів функції.

Теорема 2.2.7. *Всякий вираз булевої алгебри може бути переворений у досконалу диз'юнктивну нормальну форму.*

Доведення. Нехай $S(x_1, x_2, \dots, x_n)$ — деякий вираз булевої алгебри і $F(x_1, x_2, \dots, x_n)$ — ДНФ цього виразу. Розглянемо довільний елементарний добуток виразу F (наприклад, $x_1 \wedge x_2 \wedge \dots \wedge x_k$, $k = 1, 2, \dots, n$). Якщо $k = n$, то даний добуток є конституентою одиниці, і в цьому випадку переходимо до наступного елементарного добутку. Якщо $k < n$, то, використовуючи закон $x \vee \neg x = 1$, замінюємо $x_1 \wedge x_2 \wedge \dots \wedge x_k$ на $x_1 \wedge x_2 \wedge \dots \wedge x_k \wedge (x_{k+1} \vee \neg x_{k+1})$ і застосовуємо відповідні закони дистрибутивності. Цей процес продовжуємо доти, доки всі елементарні добутки не будуть перетворені в конституенти одиниці для множини $\{x_1, x_2, \dots, x_n\}$ змінних виразу S .

Теорема доведена.

Теорема 2.2.8. Довільна булева функція може бути представлена одною (з точністю до перестановки диз'юнктивних членів і їх множників) досконалою диз'юнктивною нормальнюю формою.

Доведення. Нехай $f(x_1, x_2, \dots, x_n)$ — довільна n -арна булева функція, а (a_1, a_2, \dots, a_n) — довільний набір з області визначення функції f , на якому вона приймає значення 1. Позначимо $K(a_1, a_2, \dots, a_n) = \tilde{x}_1 \wedge \tilde{x}_2 \wedge \dots \wedge \tilde{x}_n$ конституенту одиниці, множники якої визначаються за таким правилом:

$$\tilde{x}_i = \begin{cases} x_i, & \text{якщо } a_i = 1, \\ \neg x_i, & \text{якщо } a_i = 0. \end{cases}$$

Побудовану таким чином конституенту будемо називати конституентою, яка відповідає набору (a_1, a_2, \dots, a_n) . Очевидно, що із 2^n конституент одиниці тільки вона (з точністю до перестановки множників) буде дорівнювати 1 на наборі (a_1, a_2, \dots, a_n) . Звідси випливає, що ДНФ, яка складена з усіх конституент одиниці, що відповідають наборам, де функція f приймає значення 1, являє собою булеву функцію f . Ясно також, що всяка зміна в складі конституент приводить до ДНФ, яка вже не буде представляти цю булеву функцію.

Теорема доведена.

Наслідок 2.2.5. Всяка булева функція може бути представлена виразом булевої алгебри, тобто система операцій \vee , \wedge і \neg функціонально повна.

Підкреслимо, що згідно із законами де Моргана, функціонально повними будуть і класи функцій \vee , \neg та \wedge , \neg .

Наслідок 2.2.6. Для всяких двох виразів булевої алгебри можна перевірити їх еквівалентність.

Дійсно, якщо вирази S і S' еквівалентні, то кожний з них згідно з теоремами 2.2.6—2.2.8 перетворюється до однієї і тієї ж ДНФ. Залишається перевірити, чи це одна і та ж ДНФ, чи ні.

Слід підкреслити той факт, що ДДНФ заданої функції може являти собою досить громіздку конструкцію. Наприклад, функція $f(x, y, z) = x \vee y \vee z$ має сім наборів (крім набору $(0, 0, 0)$), на яких вона дорівнює одиниці. Отже, її ДДНФ містить сім конституент одиниці.

Принцип двоїстості. Функція $f^*(x_1, x_2, \dots, x_n) = \neg f(\neg x_1, \neg x_2, \dots, \neg x_n)$ називається двоїстою до функції $f(x_1, x_2, \dots, x_n)$. З даного означення і законів булевої алгебри випливає, що функція, двоїста до двоїстої функції, збігається з первинною, тобто $(f^*)^* = f$. Легко переконатися в тому, що диз'юнкція і кон'юнкція, а також 0 і 1 — двоїсті одна до другої функції, а функція заперечення двоїста до самої себе. Дійсно,

$$\begin{aligned} (\neg x)^* &= \neg(\neg\neg x) = \neg x; \\ (0)^* &= \neg(0) = 1 \text{ і } (1)^* = \neg(1) = 0; \\ (x \vee y)^* &= \neg(\neg x \vee \neg y) = \neg\neg x \wedge \neg\neg y = x \wedge y; \\ (x \wedge y)^* &= \neg(\neg x \wedge \neg y) = \neg\neg x \vee \neg\neg y = x \vee y. \end{aligned}$$

Теорема 2.2.9. (Закон двоїстості.) *Hexай $F(x_1, x_2, \dots, x_n) = f(f_1(x_1, x_2, \dots, x_n), f_2(x_1, x_2, \dots, x_n), \dots, f_m(x_1, x_2, \dots, x_n))$. Тоді $F^*(x_1, x_2, \dots, x_n) = f^*(f_1^*(x_1, x_2, \dots, x_n), f_2^*(x_1, x_2, \dots, x_n), \dots, f_m^*(x_1, x_2, \dots, x_n))$, де $F^*, f^*, f_1^*, f_2^*, \dots, f_m^*$ — функції, двоїсті до функцій $F, f, f_1, f_2, \dots, f_m$ відповідно.*

Доведення випливає з таких рівностей:

$$\begin{aligned} F^*(x_1, \dots, x_n) &= \neg F(\neg x_1, \dots, \neg x_n) = \\ &= \neg f(f_1(\neg x_1, \dots, \neg x_n), f_2(\neg x_1, \dots, \neg x_n), \dots, f_m(\neg x_1, \dots, \neg x_n)) = \\ &= \neg f(\neg f_1^*(\neg x_1, \dots, \neg x_n), \neg f_2^*(\neg x_1, \dots, \neg x_n), \dots, \neg f_m^*(\neg x_1, \dots, \neg x_n)) = \\ &= \neg f(\neg f_1^*(x_1, \dots, x_n), \neg f_2^*(x_1, \dots, x_n), \dots, \neg f_m^*(x_1, \dots, x_n)) = \\ &= f^*(f_1^*(x_1, \dots, x_n), f_2^*(x_1, \dots, x_n), \dots, f_m^*(x_1, \dots, x_n)). \end{aligned}$$

Теорема доведена.

Виходячи із закону двоїстості для констант 0 і 1, операцій диз'юнкції і кон'юнкції, а також заперечення, одержуємо, що в булевій алгебрі функцій вираз S^* , двоїстий до виразу S , є результатом одночасної заміни у виразі S всіх диз'юнкцій на кон'юнкції, всіх кон'юнкцій на диз'юнкції, всіх нулів на одиниці і всіх одиниць на нулі. Користуючись поняттям двоїстого виразу, можна перейти від уже розглянутого зображення булевих функцій у вигляді ДНФ до іншого відомого зображення — **кон'юнктивної нормальної форми (КНФ)**. При цьому двоїстим поняттю елементарного добутку буде поняття елементарної диз'юнкції, двоїстим поняттю конституенти одиниці буде поняття конституенти нуля.

3. Алгебра Жегалкіна. Алгебра Жегалкіна — це множина булевих функцій, на якій визначені такі операції:

- нульварна операція 1;

- бінарна операція кон'юнкція;
- бінарна операція сума за модулем 2.

Користуючись операціями додавання за модулем 2 і одиниці, можна ввести константу 0, тобто $1 + 1 = 0$.

Основними тотожними співвідношеннями даної алгебри є такі співвідношення:

- ж1) $x \wedge (y \wedge z) = (x \wedge y) \wedge z; \quad x \wedge y = y \wedge x;$
- ж2) $x + (y + z) = (x + y) + z; \quad x + y = y + x;$
- ж3) $x + x = 0;$
- ж4) $x + 0 = x;$
- ж5) $x(y + z) = xy + xz.$

Операції заперечення і диз'юнкції в цій алгебрі вводяться за допомогою таких співвідношень:

$$\begin{aligned} -x &= x + 1; \\ x \vee y &= x \wedge y + x + y. \end{aligned}$$

Користуючись цими співвідношеннями, всяку ДДНФ можна перетворити до виразу, в якому відсутні операції заперечення і диз'юнкції. Після цього, застосовуючи закони ж1–ж5 і приводячи подібні члени, одержуємо представлення булевої функції в алгебрі Жегалкіна, яке називається **поліномом Жегалкіна**. Для однозначності задання булевої функції поліномом Жегалкіна користуються так званим канонічним поліномом.

Канонічним поліномом називається скінчenna suma таких попарно різних добутків змінних, де в одному добутку жодна змінна не зустрічається більше одного разу. При цьому до числа добутків відносять добутки, які складаються як із одного співмножника (окрема змінна), так і з пустої множини співмножників (константа 1).

Приклад 2.2.10

Вирази $x \wedge y \wedge z + x \wedge z$, y , 1 , очевидно, є канонічними поліномами, а вирази $z \wedge x \wedge z$, $x \wedge z + z \wedge x$ — не є такими поліномами, оскільки в першому з них повторюється множник z , а в другому — доданки рівні між собою. ◀

Теорема 2.2.10. Всяка булева функція може бути представлена канонічним поліномом Жегалкіна, причому це представлення єдине з точністю до перестановки доданків і їх співмножників.

Доведення. Нехай $f(x_1, x_2, \dots, x_n)$ — довільна булева функція і a_1, a_2, \dots, a_m ($m = 2^n$) — така послідовність попарно різних наборів з області визначення цієї функції, що у будь-яких двох наборах, які стоять поряд у цій послідовності, число оди-

ниць у другому або таке ж, як і в першому, або на одиницю більше. Множину, яка складається з перших k ($1 \leq k \leq 2^n$) членів цієї послідовності, позначимо G_k . Побудуємо таку послідовність F_1, F_2, \dots, F_k канонічних поліномів змінних x_1, x_2, \dots, x_n , що значення $F_k(x_1, x_2, \dots, x_n)$ і $f(x_1, x_2, \dots, x_n)$ збігаються на множині G_k , і поліноми F_k і F_{k+1} або збігаються, або другий одержано з першого додаванням ще одного доданка. Якщо така послідовність побудована, то $F_m(x_1, x_2, \dots, x_n)$, очевидно, буде представляти саму функцію $f(x_1, x_2, \dots, x_n)$. Виберемо F_1 як константу $f(0, \dots, 0)$.

Припустимо, що вже побудовані перші k ($k < 2^n$) поліномів шуканої послідовності. Візьмемо набір a_{k+1} , який розрізняє між собою множини G_k і G_{k+1} , і порівняємо значення $F_k(a_{k+1})$ і $f(a_{k+1})$. Якщо вони збігаються, то за F_{k+1} може бути вибраний поліном F_k . Коли ж ці значення відрізняються одне від одного, то будемо добуток Π_{k+1} , співмножниками якого є ті і тільки ті змінні з x_1, x_2, \dots, x_n , які на наборі a_{k+1} дають одиницю, і покладемо $F_{k+1} = \Pi_{k+1} + F_k$. Для набору a_{k+1} значення початкової функції f і полінома F_{k+1} збігаються за побудовою. Для наборів із G_k значення f і F_{k+1} збігатимуться внаслідок того, що на них добуток Π_{k+1} перетворюватиметься в нуль.

Дійсно, оскільки всякий набір із G_k відмінний від a_{k+1} і містить не більше одиниць, ніж число співмножників у добутку Π_{k+1} , то для всякого такого набору хоча б один співмножник із Π_{k+1} дорівнюватиме нулю. Отже, існування канонічного полінома, який представляє функцію $f(x_1, x_2, \dots, x_n)$, доведено.

Покажемо тепер єдиність цього полінома. Нехай F_1 і F_2 — поліноми, які представляють одну і ту ж функцію f і відрізняються не тільки порядком переліку доданків і співмножників. Розглянемо поліном $F = F_1 + F_2$, в якому згідно з тотожністю $x + x = 0$ виконується зведення подібних доданків. Поліном F внаслідок припущення про вибір F_1 і F_2 повинен відрізнятися від константи 0, і крім того, через те, що F_1 і F_2 представляють одну і ту ж функцію, він повинен на всіх наборах з області визначення перетворюватися в нуль. Але ці умови взаємосуперечні. Дійсно, якщо поліном F має доданок 1, то на наборі $(0, 0, \dots, 0)$ він буде відмінним від 0, в протилежному випадку виберемо в F доданок з мінімальним числом співмножників і побудуємо набір, на якому всі доданки, крім вираного, перетворюються в нуль.

Теорема доведена.

З доведеної теореми випливає, що система булевих функцій, яка складається з константи 1, кон'юнкції і суми за модулем 2, як і розглянута вище система (див. наведений вище приклад 2), функціонально повна.

Класи Поста. Розглянемо тепер питання загального характеру. Нехай задана довільна система булевих функцій. Які умови повинна задовольняти ця система, щоб бути функціонально повною. Ці умови називаються **умовами повноти**. Пост, характеризуючи умови повноти довільної системи булевих функцій, виділив п'ять класів булевих функцій, замкнутих відносно суперпозиції:

- 1) функції, які зберігають 0;
- 2) функції, які зберігають 1;
- 3) функції, двоїсті самі собі;
- 4) лінійні функції;
- 5) монотонні функції.

Функція $f(x_1, x_2, \dots, x_n)$ називається функцією, яка зберігає 0, якщо $f(0, \dots, 0) = 0$.

Функція $f(x_1, x_2, \dots, x_n)$ називається функцією, яка зберігає 1, якщо $f(1, \dots, 1) = 1$.

Функція, двоїста сама собі, називається **самодвоїстю** функцією.

Функція називається **лінійною**, якщо кожний елементарний добуток канонічного полінома Жегалкіна, що представляє цю функцію, має не більше одного співмножника.

Функція називається **монотонною** функцією, якщо із того, що вона приймає значення 1 на деякому наборі a , випливає, що вона приймає значення 1 на всякому наборі b , який одержується з набору a шляхом заміни довільного числа нулів на одиниці.

Замкнутість кожного з п'яти класів Поста випливає з означення функцій, які входять до цих класів. Для функцій, які не є твірними розглянутих вище алгебр, належність їх до того чи іншого класу Поста наводиться в табл. 2.1.

Таблиця 2.1

Функції	\vee	\wedge	\neg	$+$
Які зберігають 0	+	+	-	+
Які зберігають 1	+	+	-	-
Самоподвійні	-	-	+	-
Лінійні	-	-	+	+
Монотонні	+	+	-	-

При \vee : + означає, що функція належить відповідному класу Поста; - — не належить.

З табл. 2.1, зокрема, видно, що кожний із п'яти класів Поста є власною підмножиною всієї множини булевих функцій.

Теорема Поста про функціональну повноту булевих функцій. Для того щоб система булевих функцій S була функціонально повною над алфавітом булевих змінних $X = \{x_1, x_2, \dots, x_n\}$, необхідно і

достатньо, щоб ця система містила хоча б одну функцію, яка не зберігає 0, хоча б одну функцію, яка не зберігає 1, хоча б одну несамодвоїсту функцію, хоча б одну нелінійну і хоча б одну немонотонну функцію.

Д о в е д е н н я. Необхідність умов теореми випливає з того, що кожний із класів Поста є власною підмножиною всієї множини булевих функцій.

Для доведення достатності зафіксуємо п'ять функцій:

- $f_1(x_1, \dots, x_n)$ ($n \geq 1$), не зберігає 0;
- $f_2(x_1, \dots, x_n)$ ($n \geq 1$), не зберігає 1;
- $f_3(x_1, \dots, x_n)$ ($n \geq 1$), несамодвоїста;
- $f_4(x_1, \dots, x_n)$ ($n \geq 1$), нелінійна;
- $f_5(x_1, \dots, x_n)$ ($n \geq 1$), немонотонна.

Підкреслимо, що деякі з цих функцій можуть збігатися. Згідно з наслідком 2.2.5 досить показати, що з вибраних п'яти функцій будується, наприклад, кон'юнкція і заперечення.

Покажемо насамперед, як побудувати константи 0 і 1 в такій алгебрі. Підставляючи в функцію f_1 змінну x , одержуємо функцію $h(x) = f_1(x, x, \dots, x)$, для якої за умовою справедлива рівність $h(0) = 1$. Якщо виявиться, що $h(1) = 1$, то $h(x)$ — функція, яка тотожно дорівнює 1. Підставляючи її замість аргументів у функцію f_2 , одержуємо другу константу — 0. Якщо ж таким шляхом не вдається одержати 0 і 1, то функція $h(x)$ являє собою заперечення.

Лема 2.2.1. Якщо система S складається з несамодвоїстої функції і функції заперечення, то серед її суперпозицій існують також обидві константи — 0 і 1.

Д о в е д е н н я. Дійсно, якщо $f(x_1, \dots, x_n)$ — несамодвоїста функція, то існує такий набір (a_1, \dots, a_n) її аргументів, що $f(a_1, \dots, a_n) = f(\neg a_1, \dots, \neg a_n)$. Але тоді для $i = 1, 2, \dots, n$, підставляючи замість змінної x_i змінну x , якщо $a_i = 1$, і її заперечення $\neg x$, якщо $a_i = 0$, одержуємо функцію $h(x)$, яка незалежно від значення змінної x буде приймати одне і те ж значення, тобто буде константою. Якщо цю константу тепер підставити у функцію заперечення, то одержимо другу константу.

Лема 2.2.2. Якщо система S складається з немонотонної функції і обох констант 0 і 1, то серед її суперпозицій існує функція заперечення.

Д о в е д е н н я. Нехай $f(x_1, \dots, x_n)$ — немонотонна функція. Тоді знайдуться такі набори a і a' , що функція $f(a) = 1, f(a') = 0$, і ті компоненти, якими ці набори розрізняються, в наборі a дорівнюють 0, а у наборі a' — 1. Неважко побудувати таку послідовність $a = a_1, a_2, \dots, a_k = a'$, що всякі два сусідні набори в цій послідовності відрізняються один від одного лише по одному

компоненту, який дорівнює нулю для першого з них і одиниці — для другого. Після цього розглянемо послідовність значень $f(a_1) = 1, f(a_2) = 0, \dots, f(a_k) = 0$ і знайдемо в послідовності наборів такі два набори a_i і a_{i+1} , що $f(a_i) = 1$ і $f(a_{i+1}) = 0$. Нехай j — номер компонента, за яким розрізняються ці набори. Підставивши в функцію $f(x_1, \dots, x_n)$ замість змінної x_j змінну x , а замість решти змінних відповідні їм константи 0 і 1 з наборів a_i і a_{i+1} , одержимо функцію $h(x)$, яка, очевидно, буде запереченням.

Для остаточного завершення доведення леми тепер необхідно показати, як можна сконструювати кон'юнкцію $x_1 \wedge x_2$. Для цього візьмемо нелінійну функцію f_4 . Її канонічний поліном містить хоча б один доданок, який складається більше ніж із двох змінних (nehay це будуть змінні x_1 і x_2). Користуючись дистрибутивністю кон'юнкції відносно додавання за модулем два, функцію f_4 можна представити у вигляді $x_1 \wedge x_2 \wedge g_1(x_3, \dots, x_n) + x_1 \wedge g_2(x_3, \dots, x_n) + x_2 \wedge g_3(x_3, \dots, x_n) + g_4(x_3, \dots, x_n)$, де g_1 тутожне не дорівнює нулю. Підставляючи відповідним чином константи замість змінних x_3, \dots, x_n , функцію f_4 перетворимо до виразу $x_1 \wedge x_2 + a \wedge x_1 + b \wedge x_2 + c$, де a, b, c — константи 0 або 1. Підставимо тепер замість x_1 функцію $x_1 + b$ ($x_1 + b = x_1$, якщо $b = 0$, або $\neg x_1$, якщо $b = 1$) і замість x_2 — функцію $x_2 + a$, одержимо функцію

$$\begin{aligned} (x_1 + b) \wedge (x_2 + a) + a \wedge (x_1 + b) + b \wedge (x_2 + a) + c = \\ = x_1 \wedge x_2 + a \wedge x_1 + b \wedge x_2 + a \wedge b + a \wedge x_1 + a \wedge b + \\ + b \wedge x_2 + a \wedge b + c = x_1 \wedge x_2 + (a \wedge b + c). \end{aligned}$$

Якщо тепер константа $a \wedge b + c = 0$, то одержана функція шукана, якщо ж $a \wedge b + c = 1$, то шуканою функцією буде її заперечення.

Лема доведена. \blacktriangleleft

10. СТРУКТУРИ

Алгебра $G = (A, \Omega) \in K(\Omega, Eq)$ називається *структурою*, якщо Ω складається з двох бінарних операцій \vee (верхня грань) і \wedge (нижня грань), а $\forall a, b, c \in A$ справедливі такі співвідношення (закони):

- C1) $a \vee a = a \wedge a = a$ — ідемпотентність;
- C2) $a \vee b = b \vee a, a \wedge b = b \wedge a$ — комутативність;
- C3) $a \vee (b \vee c) = (a \vee b) \vee c,$
 $a \wedge (b \wedge c) = (a \wedge b) \wedge c$ — асоціативність;
- C4) $a \vee (a \wedge b) = a, a \wedge (a \vee b) = a$ — поглинання.

Якщо на множині A задана лише операція \vee (\wedge), то алгебра $G(A, \Omega)$ називається **верхньою (нижньою) напівструктурою**.

Якщо алгебра являє собою напівструктуру, то на її носіях можна задати відношення часткового порядку, яке має ту властивість, що для будь-яких елементів $a, b \in A$ існує елемент $c \in A$, такий, що $a \leq c$ і $b \leq c$ ($a \geq c$ і $b \geq c$) (самі елементи a і b можуть бути непорівнянними).

Дійсно, нехай $G = (A, \Omega)$ — верхня напівструктура. Визначимо для довільних $a, b \in A$: $a \leq b \Leftrightarrow a \vee b = b$. Покажемо, що \leq — частковий порядок на A . Для цього потрібно показати, що \leq — рефлексивне, антисиметричне і транзитивне відношення.

Рефлексивність. $a \leq a \Leftrightarrow a \vee a = a$, але останнє є С1.

Антисиметричність. $a \leq b$ і $b \leq a \Rightarrow a = b$: $a \leq b \Leftrightarrow a \vee b = b$, $b \leq a \Leftrightarrow b \vee a = a$ і в силу комутативності операції \vee (С2) робимо висновок, що $a = b$.

Транзитивність. $a \leq b$ і $b \leq c \Rightarrow a \leq c$: $a \leq b \Leftrightarrow a \vee b = b$, $b \leq c \Leftrightarrow b \vee c = c \Rightarrow (a \vee b) \vee c = a \vee (b \vee c) = a \vee c = c \Rightarrow a \vee c = c$, звідки випливає, що $a \leq c$.

Таким чином, структури можна вважати частково упорядкованими множинами.

Якщо G — структура, то із С1—С4 безпосередньо випливають такі властивості.

Твердження 2.2.6. Для будь-яких елементів a, b, c із G справедливі такі співвідношення:

- (1) $a \wedge b \leq a$;
- (2) $a \leq b$ тоді і тільки тоді, коли $a \wedge b = a$;
- (3) якщо $a \leq b$ і $c \leq d$, то $a \vee c \leq b \vee d$ і $a \wedge c \leq b \wedge d$;
- (4) $(a \wedge c) \vee (b \wedge c) \leq c$.

Доведення. Співвідношення (1) випливає безпосередньо з С4. Дійсно, оскільки $a \vee (a \wedge b) = a$, то за означенням відношення \leq маємо $a \wedge b \leq a$.

Доведемо (2). Якщо $a \leq b$, то $a \vee b = b$ і тоді з С4 одержуємо $a \wedge (a \vee b) = a \wedge b = a$.

Навпаки, якщо $a \wedge b = a$, і, оскільки $b \vee (b \wedge a) = b \vee a = b$, то це значить, що $a \leq b$.

Доведемо (3). Якщо $a \leq b$ і $c \leq d$, то це значить, що $a \vee b = b$ і $c \vee d = d$. Але тоді, користуючись С3, одержуємо $(a \vee c) \vee (b \vee d) = (a \vee b) \vee (c \vee d) = b \vee d$, тобто $(a \vee c) \leq (b \vee d)$. Аналогічно одержуємо $(a \wedge c) \wedge (b \wedge d) = (a \wedge b) \wedge (c \wedge d) = a \wedge c$, тобто $(a \wedge c) \leq (b \wedge d)$ на основі (2).

Доведемо (4). На основі (1) маємо, що $a \wedge c \leq c$ і $b \wedge c \leq c$. Але тоді $(a \wedge c) \vee (b \wedge c) \leq c \vee c = c$ на основі (3) і С1.

Твердження доведене.

Інші важливі нерівності для елементів структури наведені у вправах в кінці розділу (див. вправу 20).

Часто можна зустріти і таке означення структури.

Множина A , частково упорядкована відношенням \leq , називається структурою, якщо вона задовольняє такі умови:

(i) для будь-яких елементів a, b із A існує елемент $c = a \wedge b$ — перетин елементів a і b , такий, що $c \leq a$ і $c \leq b$, причому якщо c' — деякий елемент із A , такий, що $c' \leq a$ і $c' \leq b$, то $c' \leq c$;

(ii) для будь-яких елементів a, b із A існує елемент $d = a \vee b$ — об'єднання елементів a і b , такий, що $d \geq a$ і $d \geq b$, причому якщо d' — деякий елемент із A , такий, що $d' \geq a$ і $d' \geq b$, то $d' \geq d$.

З цього означення випливає, що перетин і об'єднання в структурі A визначені однозначно, тобто всяка структура є універсальною алгеброю з двома бінарними операціями \wedge і \vee .

Нехай $a \leq b$ і $a, b \in A$. Розглянемо, чому дорівнюють однозначно визначені елементи $a \wedge b$ і $a \vee b$. Внаслідок рефлексивності відношення часткового порядку \leq таким елементом є елемент a , якщо ж $a = a \wedge b$, то $a \leq a$ і $a \leq b$. Analogічно знаходимо, що $a \vee b = b$. Отже, відношення \leq може бути задане за допомогою однієї з цих операцій. Справедливе таке твердження.

Твердження 2.2.7. Якщо A — структура, означена за допомогою умов (i), (ii), то алгебра $G = (A, \{\wedge, \vee\})$ задовольняє закони С1—С4 і, навпаки, якщо $G = (A, \{\wedge, \vee\})$ — алгебра, в якій виконуються закони С1—С4, то множина A є структурою згідно з пунктами (i), (ii) означення.

Д о в е д е н н я. Закони С1 і С2 в структурі A очевидні. Розглянемо С3. Покажемо спочатку, що коли $a \leq b$ і $c \leq d$, то $a \wedge c \leq b \wedge d$ і $a \vee c \leq b \vee d$. Дійсно,

$$\begin{aligned} a \wedge c &= (a \wedge b) \wedge (c \wedge d) \leq a \wedge b \leq b, \\ a \wedge c &= (a \wedge b) \wedge (c \wedge d) \leq c \wedge d \leq d. \end{aligned}$$

Але тоді, якщо $d' = b \wedge d$, то $d' \leq b$, $d' \leq d$ і згідно з умовою (i) $a \wedge c \leq d'$, тобто $a \wedge c \leq b \wedge d$. Analogічно доводиться і друга нерівність. Тепер, користуючись доведеними нерівностями, одержимо

$$\begin{aligned} (a \wedge b) \wedge c &\leq a \wedge b \leq a, \\ (a \wedge b) \wedge c &\leq a \wedge b \leq b, \\ (a \wedge b) \wedge c &\leq c, \end{aligned}$$

звідки $(a \wedge b) \wedge c \leq a \wedge (b \wedge c)$. Analogічно із

$$\begin{aligned} a \wedge (b \wedge c) &\leq a, \\ a \wedge (b \wedge c) &\leq b \wedge c \leq b, \\ a \wedge (b \wedge c) &\leq b \wedge c \leq c \end{aligned}$$

одержуємо $a \wedge (b \wedge c) \leq (a \wedge b) \wedge c$. Внаслідок антисиметричності відношення \leq маємо $a \wedge (b \wedge c) = (a \wedge b) \wedge c$.

Розглянемо другий із законів С4. Згідно з умовою (i) $a \wedge (a \vee b) \leq a$. Але оскільки $a \leq a$ і $a \leq a \vee b$ згідно з умовою (ii), то $a = a \wedge a \leq a \wedge (a \vee b)$, тобто другий закон поглинання виконується.

Інші деталі доведення пропонуються читачеві як прості вправи.

Твердження доведене.

Якщо в структурі існують елементи 0 (нуль) і 1 (одиниця), такі, що для всякого елемента a структури G мають місце тотожності

$$a \vee 0 = a \wedge 1 = a, a \wedge 0 = 0, a \vee 1 = 1,$$

то структура G називається структурою відповідно з нулем і одиницею.

Конкретним прикладом структури може служити булеан $\mathbf{B}(U)$ деякої множини U . Частковий порядок на $\mathbf{B}(U)$ вводиться за допомогою операції включення для підмножин, а роль операцій \vee і \wedge відіграють операції \cup і \cap .

Зауважимо, що множина $\mathbf{B}(U)$ буде структурою з нулем і одиницею, роль яких виконують відповідно пуста множина \emptyset і вся множина U .

Оскільки структура є частково упорядкованою множиною, то можна розглядати послідовності елементів $a_1, a_2, \dots, a_n, \dots$ із G , таких, що

$$a_1 < a_2 < \dots < a_n < \dots \quad (a_1 > a_2 > \dots > a_n > \dots). \quad (2.2.5)$$

Послідовність (2.2.5) називається *строго зростаючою (строго спадаючою)* або просто *зростаючим (спадаючим) ланцюгом*.

Структура $G = (A, \Omega)$ називається *структурою з умовою обриву спадаючих (зростаючих) ланцюгів*, коли всякий зростаючий (спадаючий) ланцюг обривається на скінченному індексі. Іншими словами, для всякої послідовності (2.2.5) існує такий індекс n , починаючи з якого цей ланцюг стабілізується, тобто $a_n = a_{n+1} = \dots$

Структури з умовою обриву спадаючих ланцюгів є алгебрами, які мають ряд цікавих властивостей.

Теорема 2.2.11. Якщо алгебра $G = (A, \Omega)$ — структура з умовою обриву спадаючих ланцюгів, то всяка непуста підмножина множини A має хоча б один мінімальний у цій підмножині елемент.

Доведення. Припустимо супротивне: існує деяка непуста підмножина A_1 множини A , яка не має мінімальних елементів. Позначимо по одному елементу в кожній непустій під-

множині множині A_1 , користуючись аксіомою вибору, а потім побудуємо послідовність елементів $a_1, a_2, \dots, a_n, \dots$

Першим елементом a_1 беремо елемент, позначений у множині A_1 , другим беремо елемент a_2 , строго менший a_1 в множині A_1 , тобто $a_2 \in A_1 \setminus \{a_1\}$, $a_2 < a_1$ і т.д. Згідно з припущенням про відсутність мінімального елемента в множині A_1 множина $A_1 \setminus \{a_1\}$ непуста, множина $A_1 \setminus \{a_1, a_2\}$ непуста і т.д. Побудована нами послідовність — це нескінчений строго спадаючий ланцюг, і оскільки цей ланцюг лежить у множині A , то алгебра G не може бути структурою з умовою обриву спадаючих ланцюгів.

Теорема доведена.

Теорема 2.2.12. Якщо структура $G = (A, \Omega)$ має ту властивість, що всяка непуста підмножина з A має хоча б один мінімальний елемент, то в структурі G виконується умова індуктивності.

Доведення. Припустимо, що в множині A існує деяка підмножина елементів A_1 , така, що елементи з A_1 не задовольняють властивість P . Внаслідок існування мінімальних елементів, підмножина A_1 має хоча б один мінімальний елемент a . Елемент a не може бути мінімальним для всієї множини A , бо в протилежному випадку хибним буде пункт 1 умови індуктивності (див. § 1.1). Оскільки для всіх елементів $c < a$ умова P істинна, то згідно з другим посиланням умови індуктивності $P(a)$ теж повинно бути істинним. Отримана суперечність доводить теорему.

Завдяки тому, що відношення часткового порядку двоїсте, все сказане залишається справедливим і для структур з умовою обриву зростаючих ланцюгів.

Структура G називається *дистрибутивною*, якщо Eq , крім законів С1—С4, містить закон дистрибутивності, тобто для будь-яких її елементів a, b, c виконується співвідношення

$$C5) (a \vee b) \wedge c = (a \wedge c) \vee (b \wedge c).$$

Зауважимо, що не всі структури дистрибутивні, як і не всі структури відповідають умові максимальності (мінімальності) і не обов'язково повинні мати 0 або 1, або і те, і друге разом.

Структура G називається *дедекіндою*, або *модулярною*, якщо, крім законів С1—С4 для будь-яких її елементів a, b, c , таких, що $a \leq c$, виконується модулярний закон:

$$M3) (a \vee b) \wedge c = a \vee (b \wedge c).$$

Зробимо кілька зауважень відносно дистрибутивних і дедекіндових структур.

Для дистрибутивних структур справедлiva така теорема.

Теорема 2.2.13. Структура G має такі еквівалентні властивості:

- (1) G дистрибутивна;
- (2) $(a \wedge b) \vee c = (a \vee c) \wedge (b \vee c)$ для довільних a, b, c із G ;
- (3) $(a \wedge b) \vee (b \wedge c) \vee (c \wedge a) = (a \vee b) \wedge (b \vee c) \wedge (c \vee a)$ для довільних a, b, c із G ;
- (4) якщо для деякого c із G виконуються рівності $a \vee c = b \vee c$ і $a \wedge c = b \wedge c$, то $a = b$.

Доведення. (1) \Rightarrow (2). Маємо

$$\begin{aligned} (a \vee c) \wedge (b \vee c) &= ((a \vee c) \wedge b) \vee c = \\ &= (a \wedge b) \vee (b \wedge c) \vee c = (a \wedge b) \vee c. \end{aligned}$$

Отже, виконується (2).

(2) \Rightarrow (3). Дійсно,

$$\begin{aligned} (a \wedge b) \vee (b \wedge c) \vee (c \wedge a) &= (a \vee (b \wedge c)) \vee (c \wedge a) \wedge \\ &\wedge (b \vee (b \wedge c)) \vee (c \wedge a) = (a \vee (b \wedge c)) \wedge (b \vee (c \wedge a)) = \\ &= (b \vee a) \wedge (c \vee a) \wedge (b \vee c) \wedge (b \wedge a) = \\ &= (a \vee b) \wedge (b \vee c) \wedge (c \vee a). \end{aligned}$$

(3) \Rightarrow (1). Нехай G відповідає умові (3), $a, b, c \in G$ і $a \leq c$.
Тоді

$$(a \wedge b) \vee (b \wedge c) = (a \wedge b) \vee (b \wedge c) \vee (c \wedge a) = (a \vee b) \wedge \\ \wedge (b \vee c) \wedge (c \vee a) = c \wedge (a \vee b) \vee (b \vee c) = c \wedge (a \vee b).$$

Отже, в G виконується модулярний закон. Далі, покладемо

$$u = (a \wedge b) \vee (b \wedge c) \vee (c \wedge a), v = (a \vee b) \wedge (b \vee c) \wedge (c \vee a).$$

За умовою $u = v$ і, значить, $c \wedge u = c \wedge v$. Оскільки $(a \wedge c) \vee (b \wedge c) \leq c$, то, враховуючи уже доведений модулярний закон, одержуємо

$$\begin{aligned} c \wedge u &= c \wedge ((a \wedge b) \vee (b \wedge c) \vee (c \wedge a)) = c \wedge ((b \wedge c) \vee \\ &\vee (a \wedge c)) \vee (a \wedge b \wedge c) = (a \wedge c) \vee (b \wedge c), \\ c \wedge v &= c \wedge ((a \vee b) \wedge (b \vee c) \wedge (c \vee a)) = c \wedge (a \vee b), \end{aligned}$$

що і потрібно було довести.

(1) \Rightarrow (4). Якщо $a \vee c = b \vee c$ і $a \wedge c = b \wedge c$, то, застосовуючи (1), одержуємо

$$\begin{aligned} a &= a \wedge (a \vee c) = a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c) = \\ &= (a \wedge b) \vee (b \wedge c) = (a \vee c) \wedge b = (b \vee c) \wedge b = b. \end{aligned}$$

(4) \Rightarrow (3). Пропонується читачеві як вправа.

Теорема доведена.

Умова (3) теореми 2.2.13 показує, що частково упорядкована множина відносно двоїстого відношення часткового порядку теж буде дистрибутивною структурою. Отже, до дистрибутивних

структур застосовний принцип двоїстості (згідно з яким всяке твердження залишається справедливим, якщо в ньому замінити знак \leq на двоїстий йому знак \geq).

Розглянемо тепер дедекіндова структури.

Теорема 2.2.14. Структура G має такі еквівалентні властивості:

(1) G — дедекіндова структура;

(2) $a \wedge ((a \wedge b) \vee c) = (a \wedge b) \vee (a \wedge c)$ для довільних елементів a, b, c із G ;

(3) якщо $a \geq b$, і для деякого c із G існують рівності $a \vee c = b \vee c$ і $a \wedge c = b \wedge c$, то $a = b$.

Д о в е д е н н я. Нехай G — дедекіндова структура, тобто виконується (1). Тоді для довільних $a, b, c \in G$ маємо $a \wedge b \leq a$ і, значить,

$$a \wedge ((a \wedge b) \vee c) = ((a \wedge b) \vee c) \wedge a = (a \wedge b) \vee (a \wedge c).$$

Якщо структура G дедекіндова і виконані умови (3), то

$$a = a \wedge (a \vee c) = a \wedge (b \vee c) = b \vee (a \wedge c) = b \vee (b \wedge c) = b.$$

Таким чином, із справедливості умови (1) випливають умови (2) і (3).

Якщо виконано (2) і $a \leq c$, то маємо

$$(a \vee b) \wedge c = ((a \wedge c) \vee b) \wedge c = (a \wedge c) \vee (b \wedge c) = a \vee (b \wedge c),$$

тобто з умови (2) випливає (1).

Нарешті, нехай виконується (3). Якщо $a, b, c \in G$ і $a \leq c$, то

$$a \vee b \leq (a \vee b) \wedge (c \vee b) \leq a \vee c$$

і

$$b \wedge c \leq (a \vee (b \wedge c)) \wedge b \leq (c \vee (b \wedge c)) \wedge b = b \wedge c,$$

звідки

$$((a \vee b) \wedge c) \vee b = a \vee b,$$

$$(a \vee (b \wedge c)) \vee b = a \vee b,$$

$$(a \vee b) \wedge c \wedge b = b \wedge c,$$

$$(a \vee (b \wedge c)) \wedge b = b \wedge c.$$

Оскільки $(a \vee b) \wedge c \geq a \vee (b \wedge c)$, то, застосовуючи рівності з умови (3), одержуємо $(a \vee b) \wedge c = a \vee (b \wedge c)$.

Теорема доведена.

Теорема 2.2.15. (Закон скорочення.) Якщо a, b, c — елементи дедекіндової структури і $(a \vee b) \wedge c = 0$, то $a \wedge (b \vee c) = a \wedge b$.

Доведення пропонується як проста вправа.

На завершення зауважимо, що закон, двоїстий закону модульлярності, має вигляд

$$(a \wedge b) \vee c = a \vee (b \wedge c),$$

де $a \geq c$, тобто знову виявляється модулярним законом. Таким чином, структура, двоїста дедекіндовій структурі, сама буде дедекіндовою структурою. Отже, дедекіндові структури задовільняють принцип двоїстості.

11. БАГАТООСНОВНІ АЛГЕБРИ. АЛГЕБРА АЛГОРИТМІВ ГЛУШКОВА

Крім алгебр з однією основною множиною, доводиться розглядасти алгебри, задані на кількох різних носіях. Такі алгебри називають **багатоосновними** або **багатосортними**.

Нехай $U = \{A_i, i \in I\}$, де A_i — деякі множини, а I — множина сортів — імен, що відповідають A_i . Пару $U = \{A_i, i \in I\}$ називають **комплектом** або **набором**. В односортному випадку кожній операції відповідає її арність, а в багатосортному зожною операцією пов'язують її тип. **Типом операції** ω називають послідовність елементів із I , яка має вигляд $s = (i_1, i_2, \dots, i_n; j)$, а операцією ω типу s на комплекті $U = \{A_i, i \in I\}$ — відображення $\omega : A_{i_1} \times A_{i_2} \times \dots \times A_{i_n} \Rightarrow A_j$. Якщо $n = 0$, то тип s має вигляд $s = (j)$, і тоді ω — нульарна операція, яка виділяє елемент у множині A_j .

Багатоосновною алгеброю називається пара $G = (U, \Omega)$, де U — деякий комплект, а Ω — сигнатура операцій, які визначені на комплекті U .

На багатоосновні алгебри легко переносяться розглянуті вище означення і конструкції теорії універсальних алгебр.

Нехай $U = \{A_i, i \in I\}$, $U' = \{A'_i, i \in I\}$ — два комплекти. Комплект U' включається в комплект U ($U' \subseteq U$), якщо $A'_i \subseteq A_i$ для будь-якого i з I . Можна говорити також і про теоретико-множинні операції над комплектами, які зводяться до операцій над відповідними компонентами множин з одним і тим же іменем. Наприклад,

$$U \cup U' = \{A_i \cup A'_i, i \in I\}, \text{ де } U = \{A_i, i \in I\} \text{ і } U' = \{A'_i, i \in I\}.$$

Якщо $U' \subseteq U$, то алгебра $G' = (U', \Omega)$ називається **підалгеброю** алгебри $G = (U, \Omega)$, коли комплект U' замкнений відносно довільної операції з Ω . Багатоосновні алгебри $G = (U, \Omega)$ і $G' = (U', \Omega')$ називаються алгебрами одного і того типу, якщо вони мають одну і ту ж сигнатуру, тобто $\Omega = \Omega'$, і комплекти з одним і тим же іменем, тобто $U = \{A_i, i \in I\}$ і $U' = \{A'_i, i \in I\}$.

Відображення f алгебри $G = (U, \Omega)$ в алгебру $G' = (U', \Omega')$

одного з нею типу називається **гомоморфізмом**, якщо для довільної операції $\omega \in \Omega$ типу $s = (i_1, i_2, \dots, i_n; j)$ і будь-яких елементів $a_1 \in A_{i_1}, \dots, a_n \in A_{i_n}$ виконується рівність

$$f(\omega(a_1, a_2, \dots, a_n)) = \omega(f(a_1), f(a_2), \dots, f(a_n)),$$

де $f: A_i \Rightarrow A'_i$ для всіх $i \in I$.

Якщо для кожного i з I відображення f взаємно однозначне, то воно називається **ізоморфізмом** алгебр G і G' .

Якщо $U' = \{A'_i, i \in I\}$ — довільний підкомплект комплекту U , то перетин всіх підалгебр алгебри $G = (U, \Omega)$ з непустими $A_i, i \in I$, які містять U' , є підалгеброю алгебри G , яка породжена комплектом U' . Якщо U' породжує U , то це позначають $\{U'\} = U$.

У випадку багатоосновних алгебр, як і у випадку одноосновних алгебр, дають означення множині термів. Нехай Ω — деяка множина багатосортних операцій, а $X = \{X_i, i \in I\}$ — деякий комплект.

Означення множини багатосортних термів виконується індуктивно таким чином:

- а) термами є всі елементи з X_i і всі нульарні символи операцій типу (i);
- б) якщо ω — операція типу $s = (i, j, \dots, k; l)$ і t_1, t_2, \dots, t_n — терми типу i, j, \dots, k відповідно, то $\omega(t_1, t_2, \dots, t_n)$ — терм типу l ;
- в) інших термів немає.

Багатоосновну алгебру багатосортних термів називають **абсолютно вільною багатоосновною алгеброю термів**.

Важливим прикладом багатоосновної алгебри є алгебра алгоритмів, яку запропонував В.М. Глушков у 1965 р. Перейдемо до розгляду основних понять алгебри алгоритмів.

Нехай B — деяка множина, яку будемо називати **інформаційним середовищем**. Зв'яжемо з множиною B дві алгебри: ОП і УМ, які назовемо відповідно **алгебрами операторів** і **умов**. Елементами алгебри ОП є (часткові) перетворення множини B , які називають **операторами**, а елементами алгебри УМ — (часткові) предикати, визначені на множині B і які називають **умовами**.

Крім звичайної операції множення (суперпозиції) відображень в алгебру ОП, введені ще дві операції: α -диз'юнкція і α -ітерація операторів, що зв'язують алгебри ОП і УМ.

Результатом α -диз'юнкції $(P \vee Q)$ двох операторів P і Q буде

такий оператор R , який для всякого $b \in B$ збігається з $P(b)$, коли $\alpha(b) = 1$, або з $Q(b)$, — коли $\alpha(b) = 0$, або невизначений, коли $\alpha(b)$ невизначено.

Результатом α -ітерації $\{\alpha P\}$ оператора P є оператор R , такий,

що $\forall b \in B$ $R(b)$ дорівнює першому із станів пам'яті в ряді $b, P(b), P^2(b), \dots, P^n(b), \dots$, для якого виконується умова α . Якщо такого стану немає, то результат застосування оператора R до стану b вважається невизначеним.

Операціями в алгебрі умов УМ служать операції диз'юнкції (\vee), кон'юнкції (\wedge), заперечення (\neg) і (ліве) множення умови на оператори з ОП. Для перших трьох операцій пояснення потребують лише ті ситуації, коли разом із звичайними значеннями умов — 0 і 1 зустрічаються невизначені значення h . Правила виконання операцій у цьому випадку задаються такими співвідношеннями:

$$\begin{aligned} 1 \vee h &= h \vee 1 = 1, \\ 0 \vee h &= h \vee 0 = h, \\ h \vee h &= h, \\ 1 \wedge h &= h \wedge 1 = h, \\ 0 \wedge h &= h \wedge 0 = 0, \\ h \wedge h &= h, \\ \neg h &= h. \end{aligned}$$

Зауважимо, що відповідно до наведених співвідношень вираз $h \vee \neg h$ дорівнює h , а не 1, оскільки $h \vee \neg h = h \vee h = h$.

Нехай $u \in \text{УМ}$, $P \in \text{ОП}$ — довільні умова і оператор, $b \in B$. Тоді Pu є умовою u' , такою, що $u'(b) = 1 \Leftrightarrow u(P(b)) = 1$, $u'(b) = 0 \Leftrightarrow u(P(b)) = 0$ і $u'(b) = h$, якщо $P(b)$ або $u(P(b))$ невизначене.

Якщо в алгебрі операторів і умов зафіксувати деякі системи твірних (базиси), то елементи кожної з цих алгебр можна задавати виразами, які складаються з твірних і символів операцій системи $Z = (\text{ОП}, \text{УМ})$, які описувались вище. Такі вирази називаються *регулярними*, а сама система $Z = (\text{ОП}, \text{УМ})$ — *алгеброю алгоритмів*.

Зокрема, алгебра алгоритмів дозволяє представляти програми у вигляді виразів алгебри алгоритмів і проводити над програмами формальні перетворення. Ці перетворення ґрунтуються насамперед на *тотожніх і квазитотожніх* співвідношеннях алгебри алгоритмів і деяких інших співвідношеннях.

Тотожні співвідношення алгебри алгоритмів — це співвідношення, які виконуються для всіх операторів і умов незалежно від станів інформаційного середовища B .

Квазитотожностями називаються співвідношення, що виконуються при деяких вимогах, накладених на умови і оператори, які входять до їх запису, незалежно від стану інформаційного середовища B .

Приклади 2.2.11 (тотожностей алгебри алгоритмів)

1. $\underset{u}{(P \vee Q)} = \underset{\neg u}{(Q \vee P)};$
2. $P \underset{u}{(Q \vee R)} = \underset{\neg u}{(PQ \vee PR)};$
3. $\underset{u}{(QP \vee RP)} = \underset{u}{(Q \vee R)}P;$
4. $\underset{u \& u}{((P \vee Q) \vee R)} = \underset{u \& u}{(P \vee \underset{u \& \neg u}{(Q \vee R)})};$
5. $\underset{u}{\{e\}} = e, \underset{u \& u}{\{\{P\}\}} = \underset{u}{\{P\}}.$

Приклади 2.2.12 (квазітотожностей алгебри алгоритмів)

6. $Pu = u \Rightarrow \underset{u}{(PQ \vee PR)} = \underset{u}{P(Q \vee R)};$
7. $Pu = u, PQ = QP \Rightarrow \underset{u}{P\{Q\}} = \underset{u}{\{Q\}P};$
8. $Pu = u, PQ = QP, P^2 = P \Rightarrow \underset{u}{\{PQ\}} = \underset{u}{\{QP\}} = \underset{u}{(e \vee P)\{Q\}}.$

§ 2.3. ПОВНІ СТРУКТУРИ І НАПІВКІЛЬЦЯ

У цьому короткому параграфі розглядаються алгебраїчні структури, які не є універсальними алгебрами в тому розумінні, в якому вони введені в § 2.1. Багато алгебр мають ту властивість, що петрин, об'єднання або сума визначені в них не тільки для двох (і тому згідно із законом асоціативності — для довільного скінченного числа елементів), але і для нескінченного (зліченного) числа елементів. Такі алгебри не є універсальними, але вони існують і мають застосування. Розглянемо дві з таких алгебр — *повні структури* і *напівкільця*.

1. ПОВНІ СТРУКТУРИ

Нехай A — частково упорядкована множина відношенням часткового порядку \leq . Множина A називається *повною структурою*, якщо для довільної непустої підмножини B множини A в множині A існують такі елементи c і d , для яких виконуються умови:

(а) для всіх елементів $a \in B$ $c \leq a$, і якщо існує деякий елемент c' , такий, що $c' \leq a$ для всіх a з A , то $c' \leq c$;

(б) для всіх елементів $a \in B$ $d \geq a$, і якщо існує деякий елемент d' , такий, що $d' \geq a$ для всіх a з A , то $d' \geq d$.

Однозначно визначені елементи c і d називаються відповідно *перетином* і *об'єднанням* елементів підмножини B і їх записують так: $c = \cap B$ (або $c = \bigcap_{a \in B} a$, або $c = \bigcap_{i \in I} a_i$, якщо a_i пробігає всі елементи множини B), $d = \cup B$ (або $d = \bigcup_{a \in B} a$, або $d = \bigcup_{i \in I} a_i$).

Очевидно, що нескінчена повна структура не є універсальною алгеброю, але ясно, що вона буде структурою. Крім того, очевидно також, що повна структура має нуль і одиницю — це будуть відповідно елементи $\cup A$ і $\cap A$.

Теорема 2.3.1. Якщо частково упорядкована множина A має одиницю і в ній існує перетин для будь-яких непустих її підмножин, то A є повною структурою.

Д о в е д е н н я. Необхідно довести, що довільна непуста підмножина B множини A має об'єднання.

Нехай C — множина всіх таких елементів b із A , що $b \geq a$ для всіх a із B . Множина C непуста, оскільки $1 \in C$, а значить, існує $d = \cap C$. Покажемо, що $d = \cup B$.

Дійсно, якщо $a \in B$, то для довільного b із C $a \leq b$, отже, $a \leq d$.

З іншого боку, якщо елемент $s \in A$ такий, що для будь-якого $a \in B$ $s \geq a$, то $s \in C$ і тоді $d \leq s$.

Теорема доведена.

Нехай A і A' — частково упорядковані множини відношенням часткового порядку \leq і $\phi: A \rightarrow A'$ — деяке відображення множини A в множину A' .

Відображення ϕ називається *ізотонним*, якщо з того, що $a \leq b$, випливає $\phi(a) \leq \phi(b)$. Елемент $a \in A$ називається *нерухомою точкою* ізотонного відображення ϕ множини A в себе, якщо $\phi(a) = a$.

Теорема 2.3.2 (про нерухому точку). Якщо ϕ — ізотонне відображення повної структури A в себе, тобто $\phi: A \rightarrow A$, то $\phi(a) = a$ для деякого a з A .

Д о в е д е н н я. Нехай B — множина всіх таких елементів s із A , що $s \leq \phi(s)$. Ясно, що $0 \in B$, оскільки A — повна структура і, значить, $B \neq \emptyset$. Тоді існує $a = \cap B$, причому для всякого $s \in B$ маємо $\phi(a) \geq \phi(s) \geq s$ внаслідок ізотонності відображення ϕ . Звідси випливає, що $\phi(a) \geq a$. Але тоді $\phi(\phi(a)) \geq \phi(a)$, звідки маємо, що $\phi(a) \in B$ і, значить, $\phi(a) \leq a$. Таким чином, $a \leq \phi(a) \leq a$, тобто $\phi(a) = a$, що і потрібно було довести.

На жаль, теорема, обернена до теореми про нерухому точку, не має місця, як показує такий простий приклад.

Нехай $A = \{a, b, c\}$ — трьохелементна частково упорядкована множина, де $a \leq b$ і $b \leq c$. Очевидно, що дана множина не є повною структурою, але разом з тим всяке ізотонне відображення A в A має нерухому точку.

Приклади 2.3.1 (повних структур)

- Структура $\mathbf{B}(U)$ всіх підмножин довільної множини U є повною структурою.
- Структура всіх підалгебр довільної алгебри G є повною структурою. ▲

2. ЗАМКНУТИ НАПІВКІЛЬЦЯ

Нехай на деякій непустій множині A задана сигнатура операцій Ω , яка складається з двох бінарних операцій: додавання ($+$) і множення (\cdot), а також двох нульарних операцій: 0 і 1. Дані операції мають такі властивості:

- 1) $x + (y + z) = (x + y) + z$ — асоціативність;
- 2) $x + y = y + x$ — комутативність;
- 3) $x + x = x$ — ідемпотентність;
- 4) $x + 0 = 0 + x = x$;
- 5) $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ — асоціативність;
- 6) $x \cdot 0 = 0 \cdot x = 0$;
- 7) $x \cdot 1 = 1 \cdot x = x$;
- 8) $x \cdot (y + z) = x \cdot y + x \cdot z$,
 $(y + z) \cdot x = y \cdot x + z \cdot x$ — дистрибутивність;
- 9) зліченна сума $a_0 + a_1 + a_2 + \dots$ існує, єдина і належить множині A ;
- 10) операція множення дистрибутивна відносно нескінчених зліченних сум, тобто

$$\sum_i a_i \cdot \sum_j b_j = \sum_{i,j} a_i \cdot b_j = a_1 \cdot b_1 + a_1 \cdot b_2 + \dots + a_2 \cdot b_1 + a_2 \cdot b_2 + \dots + a_3 \cdot b_1 + a_3 \cdot b_2 + \dots$$

Ця алгебра застосовується при обчисленні значень досить важливої унарної операції **замикання**, яку в напівкільцах позначають $*$. Результатом застосування цієї операції до елемента a є елемент, що має вигляд

$$a^* = \sum_{i=0}^{\infty} a^i,$$

де $a^0 = 1$, $a^1 = a$, $a^i = a \cdot a^{i-1}$.

Приклади 2.3.2

Неважко упевнитися, що наведені нижче алгебри є напівкільцями.

1. $G = (\{0, 1\}, \{\vee, \wedge, 0, 1\})$, де операції \vee і \wedge задаються таблицями:

\vee	0	1
0	0	1
1	1	1

\wedge	0	1
0	0	0
1	0	1

Властивості 1—8 легко перевірити. Щодо властивостей 9 і 10, то слід зауважити:

$$\bigvee_{i=0}^{\infty} a_i = 0 \Leftrightarrow (\forall i \in N) a_i = 0,$$

$$\bigvee_{i=0}^{\infty} a_i = 1 \Leftrightarrow (\exists i \in N) a_i = 1;$$

2. $G_2 = (\mathbf{D}^+ \cup \{0\}, \{\min, +, \infty, 0\})$, де $\mathbf{D}^+ = \{x \in \mathbf{D} \mid x > 0\}$, а $\min(a, b)$ — це менше з чисел a і b . Роль мультиплікативної одиниці відносно операції $+$ грає 0, а роль адитивної одиниці відносно операції \min — ∞ .

3. Нехай $F(X)$ — вільний моноїд слів у деякому алфавіті X , і e — пусте слово. Тоді $G_3 = (F(X), \{\cup, \cdot, \emptyset, \{e\}\})$ — напівкільце, де \cup — об'єднання множин слів, операція множення множин L і L' визначається так:

$$L \cdot L' = \{pq \mid p \in L, q \in L'\},$$

а pq — слово, одержане в результаті операції конкатенації.

Відносно законів 9 і 10 необхідно зауважити, що

$$p \in \bigcup_{i=0}^{\infty} A_i \Leftrightarrow p \in A_i$$

для деякого i із N .

4. Алгебра $G_4 = (\{0, 1\}, \{+, \cdot, 0, 1\})$, де операції $+$ і \cdot задаються таблицями:

$+$	0	1
0	0	1
1	1	0

\cdot	0	1
0	0	0
1	0	1

не є замкнутим напівкільцем, оскільки для операції $+$ не виконується закон ідемпотентності.

Для такої алгебри операція замикання не має сенсу, тому що

$$\begin{aligned} 1 + 1 + 1 + 1 + \dots + 1 + \dots &= \\ &= (1 + 1) + (1 + 1) + \dots + (1 + 1) + \dots = 0, \end{aligned}$$

і в той же час

$$1 + 1 + 1 + 1 + \dots + 1 + \dots = \\ = 1 + (1 + 1) + (1 + 1) + \dots + (1 + 1) + \dots = 1,$$

а це означає, що нескінчена зліченна сума невизначена. ◀

Якщо $G = (A, \Omega)$ — замкнуте напівкільце і $a \in A$, то $a^* = 1 + a + a^2 + \dots$. Властивість 9 гарантує, що $a^* \in A$, а із 9 і 10 випливає, що $0^* = 1^* = 1$, а також, що $a^* = 1 + a \cdot a^*$.

Наприклад, в напівкільцях, що були наведені вище як приклади 2.3.2, маємо:

- 1) $a^* = 1$ для $a = 1$ і для $a = 0$;
- 2) $a^* = 0$ для будь-якого a із A ;
- 3) $L^* = \{e\} \cup L \cup L^2 \cup L^3 \cup \dots = \{e\} \cup \{p_1 p_2 \dots p_k \mid p_1, p_2, \dots, p_k \in L, k \geq 1\}$ для всіх L із $F(X)$. Зокрема, якщо $L = \{a, b\}$, то $L^* = \{e\} \cup \{a, b\} \cup \{aa, ab, ba, bb\} \cup \{aaa, aab, abb, aba, \dots\} \dots$

Контрольні питання

1. Яке відношення називається відношенням конгруентності?
2. Дайте означення підалгебри алгебри.
3. Яке найбільше число підалгебр може мати алгебра, яка складається з чотирьох елементів?
4. Що таке абсолютно вільна алгебра?
5. Чи буде скінченою алгебра термів, в якої $X = \{x, y, a, b\}$ і $F = \{+, -, *, /\}$?
6. Дайте означення гомоморфізму (ізоморфізму) векторних просторів.
7. Чи всяка група буде напівгрупою?
8. Яка різниця між кільцем і полем?
9. Побудуйте ізоморфізм між напівгрупами $G = \{e, a, a^2, a^3, \dots\}$ і \mathbb{N} , де \mathbb{N} — множина натуральних чисел з основною операцією додавання, а основною операцією на G є множення ($a^k a^l = a^{k+l}$).
10. Наведіть приклад кільця з дільниками нуля.
11. Поясніть, чому напівкільце не є універсальною алгеброю.
12. Наведіть приклад повної структури.

Задачі і вправи

1. Наведіть доведення твердження 2.1.2.
2. Доведіть, що в групі одиничний елемент єдиний і що всякий елемент групи має єдиний обернений елемент.
3. Доведіть, що гомоморфний образ напівгрупи, групи, кільця є відповідно напівгрупою, групою, кільцем.
4. Доведіть, що всяка ненульова підгрупа адитивної циклічної групи сама є циклічною групою.

5. Доведіть, що всяка скінченна група, яка складається з n елементів, ізоморфна симетричній групі підстановок деякої множини M із n елементів.

6. Сформулюйте і доведіть теорему про гомоморфізми для напівгруп, груп, кілець.

7. Чи може вільна група мати абелеву підгрупу? Наведіть приклад.

8. Доведіть, що слова, з яких складається множина елементів вільної групи, задовольняють закон асоціативності.

9. Доведіть, що скінченна напівгрупа з одиницею і законом (лівого або правого) скорочення являє собою групоподібну.

10. Доведіть, що підгрупа H групи G індексу 2 буде нормальним дільником групи G .

11. Довизначіть таблицю додавання GNS :

+	0	1	2	3	4
0	0	1	2	3	4
1	1	3	4	2	0
2	2	4			
3	3	2			
4	4	0			

12. Поясніть, чому не можна коректно довизначити таблицю додавання

+	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	0	3	5	2	4
2	2	3				
3	3	5				
4	4	2				
5	5	4				

(щоб упевнитися, що дане визначення некоректне, знайдіть, чому дорівнює $1+3$).

13. Покажіть, що коли бінарним відношенням R і R_1 відповідають матриці $A(R)$ і $A(R_1)$, то відношенню $R \circ R_1$ відповідає матриця $A(R) \cdot A(R_1)$. Які корисні властивості можна одержати з цього?

14. Покажіть, що порядок симетричної групи n -го степеня дорівнює $n!$

15. Доведіть, що ядро $\text{ker}(h)$ гомоморфізму h групи G в групу G' є нормальним дільником групи G .

16. Чи буде гомоморфізмом відображення h із прикладу гомоморфізму абелевих груп, якщо його задати так:

$$h(n) = \begin{cases} 1, & \text{коли } n \text{ непарне,} \\ -1, & \text{коли } n \text{ парне?} \end{cases}$$

17. Наведіть доведення теореми 2.2.3.

18. Знайдіть кільця, задані наведеною нижче таблицею, де елемент 1 відіграє роль одиниці кільця

+	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	3	0	5	2	4
2	2	0				
3	3	5				
4	4	2				
5	5	4				

19. Побудуйте кільце для:

a) "трійкової" арифметики (обчислення за модулем 3):

+	0	1	2
0	0	1	2
1	1	1	0
2	2	0	1

б) "п'ятіркової" арифметики:

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

20. Доведіть, що для будь-яких елементів деякої структури G виконуються такі властивості:

- а) з $a \leq b$ і $c \leq b$ випливає $a \vee c \leq b$;
- б) з $a \leq b$ випливає, що для довільного c із G $a \vee c \leq b \vee c$;
- в) з $a \leq b$ і $a \leq c$ випливає $a \leq b \wedge c$;
- г) з $a \leq b$ випливає, що для довільного c із G $a \wedge c \leq b \wedge c$;
- д) $a \vee (b \wedge c) \leq (a \vee b) \wedge (a \vee c)$;
- е) $a \wedge (b \vee c) \geq (a \wedge b) \vee (a \wedge c)$;
- ж) з $a \leq b$ випливає $a \vee (b \wedge c) \leq (a \vee b) \wedge c$.

21. Доведіть, що всі можливі бінарні відношення, задані на деякій множині A і упорядковані відносно включення множин, утворюють повну структуру.

22. Чи будуть повними структурами відносно включення рефлексивні, симетричні, антисиметричні, транзитивні відношення, задані на деякій непустій множині A ?

23. Доведіть, що всякий ланцюг є структурою.

Розділ 3

ЕЛЕМЕНТИ ТЕОРІЇ АЛГОРИТМІВ І МАТЕМАТИЧНОЇ ЛОГІКИ

У даному розділі розглядаються поняття алгоритму і алгоритмічної системи, а також основні (класичні) числення формальної математичної логіки — числення висловлювань і числення предикатів першого порядку. На прикладі числення висловлювань розглядаються методи доведення теорем (виведення формул числення висловлювань), які потім переносяться на числення предикатів. У кінці розділу наводиться класифікація логік, яка демонструє виразність того чи іншого логічного числення.

§ 3.1. ПОНЯТТЯ АЛГОРИТМУ І АЛГОРИТМІЧНОЇ СИСТЕМИ

У цьому параграфі ми розглянемо поняття алгоритму і пов'язані з ним основні алгоритмічні системи.

1. ІНТУЇТИВНЕ ПОНЯТТЯ АЛГОРИТМУ

Інтуїтивне поняття алгоритму містить у собі кілька загальних рис, які ясно вимальовуються з попередніх прикладів алгоритмів побудови ДНФ і КНФ і часто визнаються як характерні для поняття алгоритму.

1. Алгоритм — це процес послідовної побудови величин, який проходить у дискретному часі таким чином, що в початковий момент задається початкова скінчена множина величин, а в кожний наступний момент система величин одержується за цілком визначеним законом (програмою) із системи величин, які були в попередній момент часу (*дискретність алгоритму*).

2. Система величин, що одержується в якийсь відмінний від початкового момент часу, однозначно визначається системою ве-

личин, одержаних у попередні моменти часу (*детермінованість алгоритму*).

3. Закон одержання подальших систем величин з попередніх повинен бути простим і локальним (*елементарність кроків алгоритму*).

4. Якщо спосіб одержання наступної величини з якої-небудь заданої величини не дає результату, то слід вказати, що вважається результатом алгоритму (*направленість алгоритму*).

5. Початкова система величин може вибиратись із деякої потенціально нескінченної множини (*масовість алгоритму*).

Поняття алгоритму, в деякій мірі визначене пунктами 1—5, звичайно, не строго: у формулюваннях цих пунктів зустрічаються слова *спосіб, величина, простий, локальний*, точний зміст яких не встановлено. Надалі це нестрогое поняття алгоритму буде називатися *безпосереднім або інтуїтивним поняттям алгоритму*.

Користуючись інтуїтивним поняттям алгоритму, можна описувати процес розв'язання тієї чи іншої задачі, але з його допомогою не можна впевнитися в тому, що описаний процес являє собою алгоритм. Дійсно, одна справа — довести існування алгоритму, а зовсім інша — довести його відсутність. Для цього потрібно знати точно, що таке алгоритм. Розв'язок цієї задачі одержано в середині 30-х років ХХ ст. у двох формах. Перша ґрунтувалась на понятті рекурсивної функції, а друга — на точно окресленому класі процесів. Обидві форми одержали назву *алгоритмічні системи*.

2. ОБЧИСЛЮВАНІ І ЧАСТКОВО РЕКУРСИВНІ ФУНКЦІЇ. ТЕЗА ЧЕРЧА

Нехай N — множина натуральних чисел і $F = \{f_1, f_2, \dots, f_n, \dots\}$ — сукупність n -арних функцій на множині N .

Далі часто будемо зустрічати унарні функції o , s і n -арну функцію $I_m^n(x_1, \dots, x_n)$, які визначаються таким чином:

$$\begin{aligned} o(x) &= 0, \\ s(x) &= x + 1, \\ I_m^n(x_1, \dots, x_n) &= x_m. \end{aligned}$$

Ці функції називатимемо *найпростішими функціями*. До найпростіших відносяться також і n -арну функцію $o^n(x_1, \dots, x_n)$, яка дорівнює нулю для всіх $x_1, x_2, \dots, x_n \in N$.

Розглянемо $n + 1$ функцію із F — функцію f арності n і функції f_1, \dots, f_n однієї і тієї ж арності m .

Говорять, що m -арну часткову функцію $g(x_1, \dots, x_m)$ одержано

внаслідок *операції суперпозиції* або *підстановки* (S^{n+1}) із функцій f^n, f_1^m, \dots, f_n^m , якщо для будь-яких $x_1, \dots, x_m \in \mathbf{N}$

$$g^m(x_1, \dots, x_m) = f^n(f_1^m(x_1, \dots, x_m), \dots, f_n^m(x_1, \dots, x_m)),$$

деякі з x_i можуть входити в f фіктивно.

Приклад 3.1.1

Розглянемо n -арну функцію $f(x_1, \dots, x_n) = a$, де a — константа. Тоді

$$a = \underbrace{s(\dots s}_{a \text{ разів}}(o^n(x_1, \dots, x_n)) \dots),$$

тобто константу a одержуємо за допомогою операції суперпозиції з найпростіших функцій o і s . \blacktriangleleft

Нехай задано довільні часткові функції: n -арна функція g і $n+2$ -арна функція h . Говорять, що $n+1$ -арна функція f одержана *операцією примітивної рекурсії* з функції g, h ($R(g, h)$), якщо для всіх $x_1, \dots, x_n, y \in \mathbf{N}$ маємо

$$\begin{aligned} f(x_1, \dots, x_n, 0) &= g(x_1, \dots, x_n), \\ f(x_1, \dots, x_n, y + 1) &= h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y)). \end{aligned}$$

Означення 3.1.1. Нехай Z — деяка система часткових функцій. Функція f називається *примітивно рекурсивною* відносно системи функцій Z , якщо f можна одержати з функцій системи Z із найпростіших функцій o, s, I за допомогою скінченного числа операцій підстановки і примітивної рекурсії. Примітивно рекурсивна функція f відносно пустої системи функцій Z називається *просто примітивно рекурсивною функцією*.

Зауважимо, що операції підстановки і примітивної рекурсії, будучи застосованими до повністю визначених функцій, дають знову повністю визначені функції. Зокрема, всі примітивно рекурсивні функції повністю визначені.

Приклад 3.1.2

1. Для n -арної операції $o^n(x_1, \dots, x^n) = 0$ маємо

$$o^n(x_1, \dots, x_n) = S^2(o, I_1^n) = o(I_1^n(x_1, \dots, x_n)) = o(x_1) = 0.$$

Отже, ця функція примітивно рекурсивна.

2. Функцію $f(x, y) = x + y$ можна одержати з функції $I_1^1(x)$ і $h(x, y, z) = z + 1$ операцією примітивної рекурсії. Дійсно,

$$x + 0 = x = I_1^1(x),$$

$$x + (y + 1) = (x + y) + 1 = s(x + y).$$

3. Функція $x - y$ в множині \mathbf{N} часткова. Для того щоб зробити цю функцію повністю визначеною на множині \mathbf{N} , розглядають зрізану різницю

$$x \div y = \begin{cases} x - y, & x \geq y, \\ 0, & x < y. \end{cases}$$

Безпосередньо з означення цієї функції випливають такі властивості:

$$\begin{aligned} x \div 0 &= x, \\ x \div (y + 1) &= (x \div y) \div 1, \\ (x \div y) \div z &= x \div (y + z). \end{aligned}$$

Функція $x \div 1$ примітивно рекурсивна, оскільки

$$\begin{aligned} 0 \div 1 &= 0 = o(x), \\ (x + 1) \div 1 &= x = I_1^2(x, y), \end{aligned}$$

тобто її можна одержати з найпростіших функцій o і I_1^2 операцією примітивної рекурсії.

З примітивної рекурсивності функції $x \div 1$ і наведених властивостей операцій зрізаної різниці маємо

$$\begin{aligned} x \div 0 &= x = I_1^1(x), \\ x \div (y + 1) &= (x \div y) \div 1, \end{aligned}$$

тобто функцію $x + y$ можна одержати за допомогою операції примітивної рекурсії з функцій $I_1^1(x)$ і $h(x, y, z) = z \div 1$.

Звідси також випливає, що $|x - y| = (x \div y) + (y \div x)$ — теж примітивно рекурсивна функція.

4. Функцію $f(x, y) = x \cdot y$ можна одержати з функцій $o(x)$ і $h(x, y, z) = z + x$ операцією примітивної рекурсії. Справді,

$$\begin{aligned} x \cdot 0 &= 0 = o(x), \\ x \cdot (y + 1) &= x \cdot y + x, \end{aligned}$$

оскільки $x + y$ — примітивно рекурсивна функція.

5. Аналогічно функцію x^n можна отримати з функцій $g(x) = 1$ і $h(x, y, z) = z \cdot x$ операцією примітивної рекурсії.

Дійсно,

$$\begin{aligned} x^0 &= 1, \\ x^{n+1} &= x^n \cdot x. \end{aligned}$$

Розглянемо ще деякі властивості примітивно рекурсивних функцій, які знадобляться надалі.

Теорема 3.1.1. Нехай n -арна часткова функція g примітивно рекурсивна (відносно системи часткових функцій Z). Тоді n -арні функції f , визначені за допомогою рівнянь

$$f(x_1, x_2, \dots, x_n) = \sum_{i=0}^{x_n} g(x_1, x_2, \dots, x_{n-1}, i),$$

$$f(x_1, x_2, \dots, x_n) = \prod_{i=0}^{x_n} g(x_1, x_2, \dots, x_{n-1}, i),$$

також примітивно рекурсивні (відносно Z).

Д о в е д е н и я. Із заданих в умові рівнянь випливає, що

$$f(x_1, \dots, x_{n-1}, 0) = g(x_1, \dots, x_{n-1}, 0),$$

$$f(x_1, \dots, x_{n-1}, y+1) = f(x_1, \dots, x_{n-1}, y) + g(x_1, \dots, x_{n-1}, y+1),$$

$$f(x_1, \dots, x_{n-1}, y+1) = f(x_1, \dots, x_{n-1}, y) \cdot g(x_1, \dots, x_{n-1}, y+1).$$

Отже, функції f будується за допомогою операції примітивної рекурсії з примітивно рекурсивних функцій $g(x_1, x_2, \dots, x_{n-1}, 0)$ і

$$h(x_1, x_2, \dots, x_{n-1}, y, z) = z + g(x_1, x_2, \dots, x_{n-1}, y+1),$$

$$h(x_1, x_2, \dots, x_{n-1}, y, z) = z \cdot g(x_1, x_2, \dots, x_{n-1}, y+1),$$

і тому ці функції примітивно рекурсивні.

Розглянемо ще один клас функцій, які часто зустрічаються. Нехай задані деякі функції $f_i(x_1, \dots, x_n)$, $i = 1, 2, \dots, k+1$, і вказані деякі умови (предикати) $u_j(x_1, \dots, x_n)$, $j = 1, 2, \dots, k$, які можуть бути для будь-якої n -ки чисел x_1, \dots, x_n істинними або хибними. Припустимо, що для n -ки чисел ніякі дві умови не можуть бути істинними одночасно (такі умови називаються *взаємно виключальними*).

Функція $f(x_1, \dots, x_n)$, задана схемою

$$f(x_1, \dots, x_n) = \begin{cases} f_1(x_1, \dots, x_n), & \text{коли } u_1(x_1, \dots, x_n) = 0, \\ \dots \\ f_k(x_1, \dots, x_n), & \text{коли } u_k(x_1, \dots, x_n) = 0, \\ f_{k+1}(x_1, \dots, x_n) & \text{для решти } x_1, \dots, x_n, \end{cases}$$

називається *кусково-заданою*.

Теорема 3.1.2. Нехай задані n -арні примітивно рекурсивні (відносно Z) функції f_1, \dots, f_{k+1} , u_1, \dots, u_k , де u_i — взаємно виключальні умови, $i = 1, 2, \dots, k$. Тоді кусково-задана функція

$$f(x_1, \dots, x_n) = \begin{cases} f_1(x_1, \dots, x_n), & \text{коли } u_1(x_1, \dots, x_n) = 0, \\ \dots \\ f_k(x_1, \dots, x_n), & \text{коли } u_k(x_1, \dots, x_n) = 0, \\ f_{k+1}(x_1, \dots, x_n) & \text{для решти } x_1, \dots, x_n \end{cases}$$

буде примітивно рекурсивною (відносно Z) функцією.

Дійсно, дану функцію можна подати у вигляді

$$f = f_1 \cdot (1 - u_1) + \dots + f_k \cdot (1 - u_k) + f_k + 1 \cdot (u_1 \cdot u_2 \cdot \dots \cdot u_k).$$

Внаслідок примітивної рекурсивності функцій обмеженої різниці, додавання і множення одержуємо примітивну рекурсивність функції f .

Зауважимо, що умови u_i в теоремі 3.1.2 мають вигляд $u_i = 0$. Оскільки умови $p_i = p_j$, $p_i \leq p_j$, $p_i < p_j$ еквівалентні умовам

$$|p_i - p_j| = 0, p_i + p_j = 0, 1 - (p_j \div p_i) = 0,$$

то теорема 3.1.2 залишається справедливою і в цих випадках, якщо p_i , p_j — примітивно рекурсивні функції.

Покажемо тепер, що частка від ділення x на y , тобто функція $[x / y]$, і остатча від ділення x на y , тобто функція $\text{rest}(x, y)$, — примітивно рекурсивні функції. Для цього необхідно довизначити дані функції до повністю визначених. Покладемо для всіх $x \in N$

$$[x / 0] = x, \quad \text{rest}(x, 0) = x.$$

Очевидно, що визначені так функції зв'язані тотожністю

$$\text{rest}(x, y) = x \div (y \cdot [x / y])$$

і, значить, з примітивної рекурсивності функції $[x / y]$ випливає примітивна рекурсивність функції $\text{rest}(x, y)$.

За визначенням, при $y > 0$ число $[x / y] = n$ задовольняє співвідношення $n \cdot y \leq x < (n + 1) \cdot y$. Звідси n дорівнює числу нулів у послідовності

$$1 \cdot y \div x, 2 \cdot y \div x, \dots, n \cdot y \div x, \dots, x \cdot y \div x.$$

Тому при $y > 0$ справедлива формула

$$[x / y] = \sum_{i=1}^x (1 - (i \cdot y \div x)).$$

Безпосередньою перевіркою можна впевнитися, що дана формула справедлива і при $y = 0$. Внаслідок примітивної рекурсивності функції \div і теореми 3.1.1 одержуємо, що функція $[x / y]$, а разом з нею і функція $\text{rest}(x, y)$ примітивно рекурсивні.

Нехай f — n -арна часткова функція з F ($n \geq 1$). Зафіксуємо перші $n - 1$ аргументів функції f і розглянемо рівняння $f(x_1, \dots, x_{n-1}, y) = x_n$ відносно змінної y . Припустимо, що існує деяка “механічна” процедура обчислення значень функції f , причому значення f невизначено тоді і тільки тоді, коли цей “механізм” працює нескінченно, не видаючи ніякого результату.

Щоб знайти розв'язок даного рівняння, будемо послідовно

обчислювати за допомогою нашого “механізму” значення $f(x_1, \dots, x_{n-1}, y)$ для $y = 0, 1, 2, \dots$ Найменше число a , для якого $f(x_1, \dots, x_{n-1}, a) = x_n$, позначимо $\mu_y(f(x_1, \dots, x_{n-1}, y)) = x_n$.

Зауважимо, що $\mu_y(f(x_1, \dots, x_{n-1}, y)) = x_n$ — n -арна часткова функція. Позначимо її M , і вважатимемо, що вона одержана операцією мінімізації часткової функції f .

Розглядаючи введені операції S^i , R , M , ми повинні обґрунтувати, чи насправді вони будуть операціями в нашему розумінні. Якщо відносно операцій підстановки і мінімізації це цілком очевидно, то для операції рекурсії потрібні деякі пояснення: чи для всяких функцій g і h існує функція f і чи буде вона єдиною?

Оскільки g і h — функції над множиною N , то відповідь на обидва питання ствердна. Якщо функція f існує, то з означення операції примітивної рекурсії одержуємо

$$\begin{aligned} f(x_1, \dots, x_n, 0) &= g(x_1, \dots, x_n), \\ f(x_1, \dots, x_n, 1) &= h(x_1, \dots, x_n, 0, g(x_1, \dots, x_n)), \\ &\dots \\ f(x_1, \dots, x_n, y+1) &= h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y)), \end{aligned}$$

і тому f визначена однозначно.

Основне означення. Часткова функція f називається *частково рекурсивною* відносно системи часткових функцій Z , якщо f може бути одержана з функцій системи Z і найпростіших функцій o , s , I_m^n за допомогою застосування скінченого числа операцій підстановки, примітивної рекурсії і мінімізації.

Часткова рекурсивна функція f відносно пустої системи функцій Z називається *просто частково рекурсивною функцією*.

За означенням універсальної алгебри пара

$$G = (F, \{R, M, S^2, \dots\}),$$

де F — множина всіх часткових функцій на N з будь-яким числом аргументів, є частковою алгеброю. Сукупність всіх частково рекурсивних функцій відносно якої-небудь системи функцій Z є підалгеброю алгебри G , породженою множиною функцій $Z \cup \{o, s, I_m^n\}$, $m, n = 1, 2, \dots$ Те саме можна сказати і про частково рекурсивні функції.

Алгебра $G_{np} = (F, \{R, S^2, S^3, \dots\})$ є частковою алгеброю. Сукупність всіх часткових примітивно рекурсивних функцій відносно якої-небудь системи функцій $Z = F$ є підалгеброю алгебри G_{np} , породженою $Z \cup \{o, s, I_m^n\}$, $m, n = 1, 2, \dots$

Важливість розглянутих класів функцій полягає у їх зв’язку з класом обчислюваних функцій. Це формулюється у вигляді таких тез.

Теза Черча. Клас алгоритмічно, або машинно обчислених часткових числових функцій збігається з класом всіх частково рекурсивних функцій.

Більш загальною є теза Тьюрінга.

Теза Тьюрінга. Клас функцій, які алгоритмічно обчислюються відносно системи функцій Z , збігається з класом частково рекурсивних функцій відносно системи Z .

Оскільки поняття обчислюваної функції не має точного означення, то довести ці тези неможливо. Але завдяки їм стало можливим придати необхідну точність формулюванням алгоритмічних проблем.

3. АЛГОРИТМІЧНІ ПРОБЛЕМИ

Під алгоритмічною проблемою будемо розуміти питання обчислюваності деяких спеціальним чином побудованих функцій. Внаслідок тез Черча і Тьюрінга питання про обчислennість функції рівнозначне питанню про її рекурсивність. Поняття рекурсивної функції строгое, і деколи за допомогою звичайної математичної техніки можна безпосередньо довести, що функція, яка вирішує проблему, не може бути рекурсивною. Якщо для розв'язку проблеми не існує рекурсивної функції, то проблема називається (алгоритмічно) *нерозв'язною*. Коли ж рекурсивна функція існує, то проблема називається (алгоритмічно) *розв'язною*.

Всякий алгоритм, як про це було сказано вище, являє собою спосіб розв'язку деякої *масової* проблеми, тобто проблеми переробки (обчислення) не одного вхідного слова (одних вхідних даних), а цілої множини вхідних слів (даних) у відповідні їм вихідні слова. Детальний аналіз задач показує, що існують такі класи задач, для розв'язку яких немає і не може бути єдиного універсального способу, тобто ці задачі належать до алгоритмічно нерозв'язних проблем. Але це зовсім не означає неможливість розв'язку будь-якої конкретної задачі з цього класу. Мова йде про неможливість розв'язку всіх задач даного класу одним і тим же способом.

Для кращого розуміння поняття алгоритмічно розв'язної і алгоритмічно нерозв'язної проблем розглянемо приклади.

Проблема тотожності в елементарній алгебрі. Для простоти обмежимося випадком, коли вирази будуються з раціональних чисел і букв за допомогою дій додавання, віднімання і множення. З шкільного курсу математики добре відомий такий спосіб розв'язання вказаної проблеми: використовуючи дистрибутивний

закон для множення, розкривають дужки в лівій і правій частинах даної тотожності і виконують зведення подібних членів. Після цього як ліва, так і права частини тотожності перетворюються в поліноми. Тотожність буде справедливою, якщо ці поліноми тотожно рівні між собою. Іншими словами, справедливість тотожності означає, що після перенесення всіх членів перетвореної таким чином тотожності в одну частину ці члени взаємно знищуються, в результаті чого тотожність перетворюється в тривіальну тотожність $0 = 0$.

Отже, проблема тотожності в елементарній алгебрі алгоритмічно розв'язна — існує єдиний конструктивний спосіб, який дає можливість за скінченне число кроків з'ясувати, чи є всяке задане співвідношення тотожністю.

Можна побудувати приклади таких алгебраїчних систем, в яких проблема тотожності нерозв'язна. До таких систем належать напівгрупи і групи, задані за допомогою системи твірних та співвідношень, які їх визначають. Вперше такі приклади для напівгруп були знайдені Е. Постом [51, 52], а для груп — П.С. Но віковим [54]. Оскільки ці проблеми формулюються майже однаково як для напівгруп, так і для груп, розглянемо лише одну з них.

Проблема тотожності для напівгруп. Нехай $F(X)$ — вільна напівгрупа, де $X = \{x_1, x_2, \dots, x_n\}$, операцією множення в якій є конкатенація, а пусте слово e відіграє роль одиниці. У цій напівгрупі можна ввести довільну множину співвідношень S , які є формальними рівностями між двома неоднаковими словами: $p_i = q_i$ ($i = 1, 2, \dots$). Два слова в напівгрупі $F(X)$ тотожні відносно заданої системи співвідношень S , якщо одне з них може бути одержане з другого в результаті довільного числа підстановок у друге слово правих частин співвідношень із S замість лівих і, навпаки, лівих замість правих. Наприклад, нехай $X = \{x, y\}$, а $S = (xy = ux)$. Тоді слова $p = xxy$ і $q = uxx$ тотожні, оскільки перше слово може бути одержане з другого внаслідок двох підстановок описаного вище типу: $q = uxx \rightarrow xux \rightarrow xxy = p$. При підстановках у зворотному порядку слово q можна одержати із слова p , що дає можливість перетворення як слова q в слово p , так і слова p в слово q .

Проблема тотожності слів для напівгруп. Нехай у довільній вільній напівгрупі $F(X)$ із скінченним числом твірних задана скінчена система співвідношень S . Необхідно знайти єдиний конструктивний спосіб, який дозволяє за скінченне число кроків визначити, чи являються будь-які два слова p і q із $F(X)$ тотожні відносно системи співвідношень S .

Для деяких систем співвідношень S сформульована проблема

розв'язна, але, як показав Пост, існують і такі системи співвідношень S , для яких проблема тотожності слів напівгруп алгоритмічно нерозв'язна.

4. МАШИНИ ПОСТА І МАШИНИ ТҮЮРІНГА

Як уже було сказано на початку даного розділу, рекурсивні функції не є єдиною алгоритмічною системою. Існує ще багато інших еквівалентних їй алгоритмічних систем: система нормальних алгорифмів Маркова, система Поста, система Тьюрінга, система Колмогорова–Успенського та ін. Розглянемо детальніше лише три — машини Поста, машини Тьюрінга і нормальні алгорифми Маркова.

Алгоритмічна система, запропонована в 1936 р. Е. Постом і названа машинами Поста, це — спеціальний клас алгоритмів. Для машин Поста вхідна і вихідна інформації задаються в алфавіті $X = \{0, 1\}$, а алгоритм — у вигляді скінченного упорядкованого набору правил (команд), які називаються *наказами*. Для запису вхідної, вихідної і проміжної інформації служить гіпотетична нескінченнна в обидві сторони *інформаційна стрічка*, поділена на окремі клітинки, в кожній з яких можна розмістити лише один символ: 0 або 1. Ті клітинки, в яких записані одиниці, називаються *відміченими*, а ті, в яких записані нулі, — *невідміченими*.

Робота алгоритму проходить дискретними кроками, на кожному виконується один із наказів, з яких складається алгоритм. Кожному такому кроку відповідає цілком визначена *активна* клітинка на інформаційній стрічці. Для першого наказу алгоритму фіксується як активна деяка *початкова* клітинка. Подальші зміни місцезнаходження активної клітинки на стрічці повинні передбачатися в самому алгоритмі. Накази, з яких складається алгоритм, можуть належати до одного з таких шести типів.

Перший тип. Відмітити активну клітинку (записати в ній одиницю) і перейти до виконання i -го наказу (i може бути довільним числом із чисел, які використані для нумерації наказів алгоритму).

Другий тип. Стерти відмітку активної клітинки (записати в ній нуль) і перейти до виконання i -го наказу.

Третій тип. Змістити активну клітинку на один крок вправо і перейти до виконання i -го наказу.

Четвертий тип. Змістити активну клітинку на один крок вліво і перейти до виконання i -го наказу.

П'ятий тип. Якщо активна клітинка відмічена (якщо в ній записана одиниця), то перейти до виконання j -го наказу, а якщо

не відмічена (якщо в ній записаний нуль), — до виконання i -го наказу.

Шостий тип. Зупинка, закінчення роботи алгоритму.

Алгоритми, які складаються із скінченного числа наказів поперелічених типів, називаються *алгоритмами Поста*. Отже, для того щоб задати машину Поста, необхідно задати таку четвірку: $X = \{0, 1\}$, активну клітинку, $\#$ — спеціальний пустий символ, P — алгоритм.

Робота машини Поста полягає у виконанні наказів алгоритму функціонування машини. Машина зупиняється тоді і тільки тоді, коли останнім виконаним наказом машини є наказ шостого типу і результатом її роботи є слово q , яке записане на інформаційній стрічці і яке називається заключним. Отже, машина Поста переробляє вхідні слова у заключні, тобто обчислює значення деякої словесної функції f_P в алфавіті X , для якої вхідні слова є її аргументами, а заключне слово — значенням цієї функції. Якщо машина Поста зупиняється з деяким словом на інформаційній стрічці, то функція, що визначена цією машиною, вважається визначеною. А якщо машина Поста не зупиняється, то значення відповідної функції вважається невизначеним. При інтерпретації заключного слова як значення функції пусті символи $\#$ ігноруються.

Доведено, що алгоритм Поста обчислює деяку частково рекурсивну функцію, і, навпаки, всяка частково рекурсивна функція може бути обчислена відповідним алгоритмом Поста. Інакше кажучи, має місце таке твердження.

Твердження. Клас всіх алгоритмів, еквівалентних алгоритмам Поста, збігається з класом всіх частково рекурсивних функцій [16, 51, 59].

Приклад 3.1.3 (приклад алгоритму Поста)

Покажемо, що найпростіша функція $S(n) = n + 1$ обчислюється на машині Поста. Для цього подамо число n в двійковій системі числення, тобто в алфавіті $X = \{0, 1\}$, запишемо його на інформаційну стрічку і встановимо активну клітинку, в якій міститься перший символ числа (його старший двійковий розряд). Тоді алгоритм P функціонування машини Поста має такий вигляд.

1. Якщо в активній клітинці 0, то перейти до п. 2, інакше перейти до п. 3.
2. Змістити активну клітинку на один крок праворуч і перейти до п. 1.
3. Якщо в активній клітинці 1, то перейти до п. 2, інакше перейти до п. 4.

4. Змістити активну клітинку на один крок праворуч і перейти до п. 5.

5. Якщо в активній клітинці 0, то перейти до п. 6, інакше, перейти до п. 7.

6. Записати в активну клітинку 1 і виконати наказ СТОП.

7. Записати в активну клітинку 0 і перейти до п. 8.

8. Змістити активну клітинку на один крок ліворуч і перейти до п. 9.

9. Якщо в активній клітинці 0, то перейти до п. 6, інакше, перейти до п. 10.

10. Якщо в активній клітинці #, то перейти до п. 6, інакше, перейти до п. 7.

Тестування цього алгоритму пропонується виконати як вправу. ▶

До описаної алгоритмічної системи Поста дуже близька система Тьюрінга, яку називають **машинами Тьюрінга** (рис.3.1.1). У цій системі також виконується запис інформації на нескінченій в обидва боки стрічці, розбитій на клітинки. Але, на відміну від системи Поста, тут для запису інформації використовується довільний скінчений алфавіт $X = \{x_1, x_2, \dots, x_n\}$.

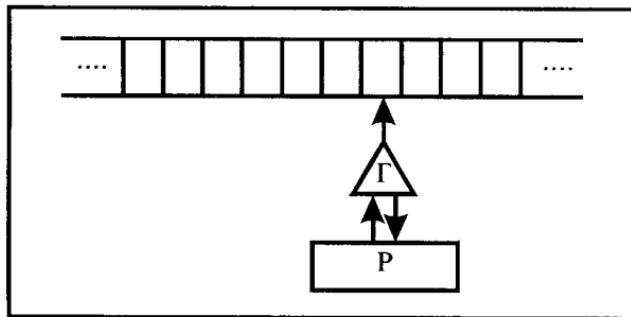


Рис. 3.1.1. Схема машини Тьюрінга

Кожна клітинка інформаційної стрічки служить для запису лише однієї букви. Цю букву можна оглядати спеціальним чутливим елементом — так званою *головкою* машини Тьюрінга, яка має змогу пересуватися вздовж інформаційної стрічки в обидва боки. Головка машини Тьюрінга може знаходитись у скінченній кількості різних станів q_1, q_2, \dots, q_m , друкувати в клітинці, яку оглядає головка, будь-яку букву x_1, x_2, \dots, x_m і залишатись на місці або переміщуватися праворуч чи ліворуч вздовж інформаційної стрічки на одну клітинку.

Задати машину Тьюрінга означає задати таку п'ятірку: X , Q , q_0 , $\#$, P , де:

X — алфавіт машини;

Q — скінчена непуста множина станів машини ($X \cap Q = \emptyset$);

q_0 — початковий стан машини ($q_0 \in Q$);

$\#$ — спеціальний “пустий” символ, такий, що $\# \notin X \cup Q$;

P — програма роботи машини.

Програма роботи машини — це скінчена множина п'ятірок вигляду $xqyq's$. Кожна така п'ятірка називається командою. Виконання команди $xqyq's$ означає, що коли головка машини Тьюрінга знаходиться в стані q і читає записану в клітинці стрічку букву x , то вона записує на місці цієї букви нову букву y (яка може збігатися з x), переходить у стан q' (який може збігатися зі станом q) і переміщується вздовж стрічки на величину $s = 0, \pm 1$.

Робота машини Тьюрінга полягає в повторенні такого циклу дій, які є спільними для будь-якої машини Тьюрінга:

а) читання символу з клітинки, яку оглядає головка;

б) пошук застосованої команди, тобто пошук команди $xqyq's$, в якій q — стан головки в даний момент часу, а x — символ у клітинці, яку оглядає головка;

в) виконання знайденої команди.

При цьому вважається, що в програмі не існує таких двох команд, які б мали однакові перші два символи.

Машина Тьюрінга зупиняється в тому і тільки в тому випадку, коли жодну з команд її програми не можна застосувати. Результатом роботи машини після її зупинки є слово, яке записане на стрічці. Отже, машина Тьюрінга стає алгоритмічною системою. При цьому вона обробляє заздалегідь записане на стрічці слово або нескінченно довго, або зупиняється після скінченного числа кроків. У першому випадку вважається, що алгоритм, який виконується машиною, не застосовний до вхідного слова p , а в другому випадку слово, яке залишається на стрічці після зупинки машини, береться як вихідне, що є результатом перетворення машиною заданого вхідного слова p .

Для машин Тьюрінга, як і для машин Поста, має місце така теорема.

Теорема 3.1.3. *Всяка частково рекурсивна функція може бути реалізована за допомогою деякої машини Тьюрінга, і, навпаки, всяка машина Тьюрінга реалізує деяку частково рекурсивну функцію.*

У розділі 7 ми ще повернемося до поняття машини Тьюрінга.

Приклад 3.1.4

1. Нехай $(X = \{0, 1\}, Q = \{q_0, q_1, q_2\}, q_0, \#, P)$, де P має такий вигляд:

$$\begin{array}{cccccc} 1 & q_0 & 1 & q_0 & 1 \\ 0 & q_0 & 0 & q_0 & 1 \\ \# & q_0 & \# & q_1 & -1 \\ 1 & q_1 & 0 & q_1 & -1 \\ 0 & q_1 & 1 & q_2 & -1 \\ \# & q_1 & 1 & q_2 & -1, \end{array}$$

де в початковий момент головка оглядає перший символ слова $p \in F(x)$ в стані q_0 .

Неважко переконатися в тому, що ця машина Тьюрінга реалізує алгоритм додавання одиниці до числа p , представленого в двійковій системі числення. Розглянемо деякі приклади. Нехай $p = 100111$; тоді маємо послідовність команд:

$$\begin{aligned} 1 & q_0 & 1 & q_0 & 1 \rightarrow 0 & q_0 & 0 & q_0 & 1 \rightarrow 0 & q_0 & 0 & q_0 & 1 \rightarrow 1 & q_0 & 1 & q_0 & 1 \rightarrow \\ \rightarrow & 1 & q_0 & 1 & q_0 & 1 \rightarrow 1 & q_0 & 1 & q_0 & 1 \rightarrow \# & q_0 & \# & q_1 & -1 \rightarrow 1 & q_1 & 0 & q_1 & -1 \rightarrow \\ \rightarrow & 1 & q_1 & 0 & q_1 & -1 \rightarrow 1 & q_1 & 0 & q_1 & -1 \rightarrow 0 & q_1 & 1 & q_2 & -1. \end{aligned}$$

Внаслідок цього слово $p = 100111$ “переписалось” машиною в слово $p' = 101000 = 100111 + 1$ в двійковій системі числення.

Нехай тепер $p = 1111$. Тоді машина виконає таку послідовність команд:

$$\begin{aligned} 1 & q_0 & 1 & q_0 & 1 \rightarrow 1 & q_0 & 1 & q_0 & 1 \rightarrow 1 & q_0 & 1 & q_0 & \rightarrow 1 & q_0 & 1 & q_0 & 1 \rightarrow \\ \rightarrow & \# & q_0 & 1 & q_1 & -1 \rightarrow 1 & q_1 & 0 & q_1 & -1 \rightarrow 1 & q_1 & 0 & q_1 & -1 \rightarrow 1 & q_1 & 0 & q_1 & -1 \rightarrow \\ \rightarrow & 1 & q_1 & 0 & q_1 & -1 \rightarrow \# & q_1 & 1 & q_2 & -1. \end{aligned}$$

Внаслідок слово $p = 1111$ “переписалось” машиною в слово $p' = 10000 = 1111 + 1$ в двійковій системі числення.

2. Нехай $(X = \{0, 1\}, Q = \{q_0, q_1\}, q_0, \#, P)$, де P має такі команди:

$$\begin{array}{c} 0 & q_0 & 0 & q_1 & -1 \\ 1 & q_0 & 1 & q_1 & -1 \\ \# & q_0 & 1 & q_1 & -1. \end{array}$$

Дія цієї машини, як неважко переконатися, полягає в тому, що, почавши роботу з будь-якого непустого слова p , до цього слова приписується зліва символ 1. Машина при цьому не зупиняється. Отже, результат роботи машини не визначений, оскільки машина не застосовна до будь-якого непустого слова $p \in F(x)$.

3. Нехай $(X = \{0, 1\}, Q = \{q_0, q_1\}, q_0, \#, P)$, де P включає такі команди:

$$\begin{array}{c} \# & q_0 & \# & q_0 & 1 \\ 0 & q_0 & 0 & q_0 & 1 \\ 1 & q_0 & 1 & q_1 & 0. \end{array}$$

Дія цієї машини Тьюрінга полягає в тому, що в слові p машина шукає символ 1 (якщо він є) і, знайшовши його, зупиняється.

Якщо слово p є записом числа в двійковій системі числення, то машина зупиняється, якщо це число не дорівнює нулю. В протилежному випадку робота машини продовжується нескінченно. ▲

5. АЛГОРИТМІЧНА СИСТЕМА МАРКОВА

Алгоритмічна система Маркова будується майже за тими самими принципами, що й попередні системи Поста і Тьюрінга, але на відміну від них вона має дещо простіший і інтуїтивно зрозуміліший характер [52, 53].

Нехай X — деякий скінчений алфавіт, $F(X)$ — напівгрупа слів у цьому алфавіті і $e \in F(X)$, де e — пусте слово. Якщо $p, q \in F(X)$, то вирази $p \rightarrow q$ і $p \rightarrow .q$ називаються **формулами підстановки** в алфавіті X . При цьому вважається, що символи \rightarrow і $.$ не належать алфавіту X , а слова p і q можуть бути пустими.

Формула підстановки $p \rightarrow q$ називається **простою**, а формула підстановки $p \rightarrow .q$ — **заключною**.

Нехай вираз $p \rightarrow [.]q$ означає будь-яку із формул підстановки $p \rightarrow q$ або $p \rightarrow .q$. Скінчена послідовність R формул підстановки в алфавіті X

$$\begin{cases} p_1 \rightarrow []q_1 \\ p_2 \rightarrow []q_2 \\ \dots \\ p_l \rightarrow []q_l \end{cases}$$

називається **схемою** або **системою переписування**. Всяка система переписування є функція $f: F(X) \rightarrow F(X)$, значення якої обчислюються за такими правилами.

1. Якщо жодне із слів p_i ($i = 1, 2, \dots, l$) не є підсловом слова p , то p залишається без змін і є результатом переписування. Цей факт будемо записувати у вигляді $R: p!$

2. Якщо серед слів p_1, p_2, \dots, p_l існують такі, що є підсловами слова p , то нехай m — таке найменше число, що $1 \leq m \leq l$ і p_m — підслово слова p . Слово p' , одержане підстановкою слова q_m замість самого лівого входження (першого входження) слова p_m в слово p , будемо позначати $R: p \dashv p'$.

Робота за такими правилами може закінчитися з двох причин:

а) якщо формула підстановки $p_m \rightarrow [.] q_m$ заключна;

б) якщо існує така послідовність r_0, r_1, \dots, r_k слів із $F(X)$, що $p = r_0, q = r_k$ і $R: r_i \vdash r_{i+1}$ для $i = 1, 2, \dots, k-2$ і або $R: r_k !$, або $R: r_{k-1} \rightarrow .r_k$.

Функція $f: F(X) \rightarrow F(X)$, визначена таким чином, називається **нормальним алгорифмом Маркова в алфавіті X** .

Робота алгорифму R може бути описана так. Нехай $p \in F(X)$. Знаходимо в R першу формулу підстановки $p_m \rightarrow [.] q_m$, таку, що p_m — підслово слова p . Виконуємо заміну першого входження слова p_m словом q_m в слові p . Якщо p' — результат цієї підстановки, то коли $p_m \rightarrow [.] q_m$ заключна, то робота алгорифму закінчується і результатом є слово p' . Якщо формула підстановки $p_m \rightarrow [.] q_m$ проста, то до слова p' застосовується той же пошук, що і до слова p , і т.д. Якщо, нарешті, одержано таке слово r_i , що $R: r_i !$, тобто ні одне із слів $p_i, i = 1, 2, \dots, l$, не є підсловом слова r_i , то робота алгорифму закінчується і r_i буде його значенням.

Із сказаного вище випливає, що можлива ситуація, коли описаний процес ніколи не закінчиться. У цьому випадку будемо говорити, що алгорифм R не застосовний до слова p .

Приклад 3.1.5

1. Нехай $X = \{a, b\}$, а R має вигляд

$$\begin{cases} a \rightarrow .e \\ b \rightarrow b. \end{cases}$$

Алгорифм R переписує всяке слово p із $F(X)$, яке має хоча б одну букву a , в слово, одержане з p шляхом викреслювання першого входження букви a в слово p . Ясно також, що $R(e) = e$ і R не застосовний до непустих слів із $F(X)$, які не мають входження букви a .

2. Нехай $X = \{x, y, x^{-1}, y^{-1}\}$, а R має вигляд

$$\begin{cases} xx^{-1} = e \\ x^{-1}x = e \\ yy^{-1} = e \\ y^{-1}y = e. \end{cases}$$

Алгорифм R переписує всяке слово p із $F(X)$ в слово q , яке не має підслів виду $xx^{-1}, x^{-1}x, yy^{-1}, y^{-1}y$.

Наприклад, слово $xxuyy^{-1}x^{-1}y^{-1}yx$ переписується таким чином: $xxuyy^{-1}x^{-1}y^{-1}yx \rightarrow xxuxx^{-1}y^{-1}yx \rightarrow xxuy^{-1}yx \rightarrow xxux$.

3. Якщо $X = \{x_1, x_2, \dots, x_n\}$, а система R має вигляд

$$\begin{cases} x_1 = e \\ x_2 = e \\ \dots \\ x_n = e, \end{cases}$$

то алгорифм R переписує всяке слово p із $F(X)$ в пусте слово e . \blacktriangleleft

Нехай R і Q — нормальні алгорифми над алфавітом X і $p \in F(X)$. Запис $R(p) \approx Q(p)$ означає, що або обидва алгорифми R і Q не застосовні до слова p , або обидва застосовні, і $R(p) = Q(p)$. Два алгорифми R і Q над алфавітом X називаються **повністю еквівалентними**, якщо $\forall p \in F(X) \quad R(p) \approx Q(p)$. Алгорифми R і Q називаються **еквівалентними відносно алфавіту X** , якщо $R(p) \approx Q(p)$ щоразу, коли $p \in F(X)$, і хоча б одне із слів $R(p)$ або $Q(p)$ визначене і теж належить $F(X)$.

Нехай $X = \{1\}$, а $X' = \{1, *\}$. Тоді всяке натуральне число n можна записати у вигляді слова \bar{n} в алфавіті X' . Дійсно, це можна зробити за допомогою такої відповідності:

$$\bar{0} = 1, \bar{1} = 11, \bar{2} = 111, \bar{3} = 1111, \bar{4} = 11111, \dots .$$

Поставимо у відповідність всякому вектору (n_1, n_2, \dots, n_k) , де n_1, n_2, \dots, n_k — натуральні числа, слово в алфавіті X' виду $\overline{n_1 * n_2 * \dots * n_k}$, яке позначимо $(\overline{n_1, n_2, \dots, n_k})$. Наприклад, вектору $(4, 0, 2)$ відповідає слово $11111*1*111$.

Нехай $f: \mathbb{N}^k \rightarrow \mathbb{N}$ — деяка часткова функція, і R_f означає алгорифм в алфавіті X' , такий, що

$$R_f(\overline{k_1, k_2, \dots, k_n}) = f(k_1, k_2, \dots, k_n)$$

тоді і тільки тоді, коли хоча б одна з частин цієї рівності визнана. При цьому вважається, що R_f не застосовний до слів, відмінних від слів виду $(\overline{k_1, k_2, \dots, k_n})$.

Функцію f будемо називати **частково обчислюваною за Марковим**, якщо існує нормальний алгорифм Q над X' , повністю еквівалентний R_f над алфавітом X' . Якщо функція f повністю визнана, то її називають просто **обчислюваною за Марковим**.

Теорема 3.1.4. Найпростіші функції $o(x) = 0$, $s(x) = x + 1$ і $I_m^n(x_1, x_2, \dots, x_n) = x_m$ обчислювані за Марковим.

Доведення зводиться до побудови відповідних нормальних алгорифмів.

1. Неважко переконатися, що функцію $o(x)$ реалізує такий алгорифм R_0 :

$$\begin{cases} * \rightarrow * & (\text{а}) \\ \alpha 11 \rightarrow \alpha 1 & (\text{б}) \\ \alpha 1 \rightarrow .1 & (\text{в}) \\ e \rightarrow \alpha, & (\text{г}) \end{cases}$$

який можна застосовувати до всіх слів в алфавіті X' , де $\alpha \notin X'$.

Дійсно, нехай $p = 11\dots11$ — довільне слово в алфавіті X . Тоді згідно з формuloю підстановки (г) маємо $R_0: p \vdash \alpha p$.

Після цього, застосовуючи формулу підстановки (б), одержуємо $R_0: \alpha p \vdash \alpha 1$ і, нарешті, застосовуючи заключну формулу підстановки (в), маємо $R_0: \alpha 1 \vdash 1$. Оскільки 1 представляє число $\bar{0}$ в алфавіті X , то R_0 і є шуканим алгорифмом внаслідок довільності слова p .

2. Для функції $s(x)$ таким алгорифмом буде алгоритм R_s виду

$$\begin{cases} * \rightarrow * & (\text{а}) \\ \alpha 1 \rightarrow .11 & (\text{б}) \\ e \rightarrow \alpha, & (\text{в}) \end{cases}$$

де $\alpha \notin X'$. Цей алгорифм застосовний тільки до слів в алфавіті X' , які є натуральними числами, причому $R_s(n) = n + 1$ для будь-якого $n \in \mathbb{N}$.

3. Більш складну структуру має алгорифм обчислення функції I_m^n . Нехай $\alpha_1, \alpha_2, \dots, \alpha_{2n} \notin X'$ і $1 \leq m < n$. Тоді позначимо Sub_m список формул підстановки

$$\begin{cases} \alpha_{2m-1}* \rightarrow \alpha_{2m-1}* \\ \alpha_{2m-1}1 \rightarrow \alpha_{2m}1 \\ \alpha_{2m}1 \rightarrow \alpha_{2m} \\ \alpha_{2m}* \rightarrow \alpha_{2m+1}. \end{cases}$$

Тепер нормальний алгорифм, який обчислює функцію I_m^n , буде мати такий вигляд:

для $m = 1$	для $1 \leq m < n$	для $m = n$
$\alpha_1^* \rightarrow \alpha_1^*$	Sub_1	
$\alpha_1 1 \rightarrow \alpha_2 1$	Sub_2	
$\alpha_2 1 \rightarrow 1\alpha_2$	
$\alpha_2^* \rightarrow \alpha_3$	Sub_{m-1}	
Sub_2	$\alpha_{2m-1}^* \rightarrow \alpha_{2m-1}^*$	
Sub_3	$\alpha_{2m-1} 1 \rightarrow \alpha_{2m} 1$	
.....	$\alpha_{2m} 1 \rightarrow 1\alpha_{2m}$	
.....	$\alpha_{2m}^* \rightarrow \alpha_{2m+1}$	
Sub_{n-1}	Sub_{m+1}	
$\alpha_{2n-1}^* \rightarrow \alpha_{2n-1}^*$	
$\alpha_{2n-1} 1 \rightarrow \alpha_{2n} 1$	Sub_{n-1}	
$\alpha_{2n} 1 \rightarrow \alpha_{2m}$	$\alpha_{2n-1}^* \rightarrow \alpha_{2n-1}^*$	
$\alpha_{2n}^* \rightarrow \alpha_{2m}^*$	$\alpha_{2n-1} 1 \rightarrow \alpha_{2n} 1$	
$\alpha_{2n} \rightarrow .e$	$\alpha_{2n} 1 \rightarrow 1\alpha_{2m}$	
$e \rightarrow \alpha_1.$	$\alpha_{2n}^* \rightarrow \alpha_{2m}^*$	
	$\alpha_{2n} \rightarrow .e$	
	$e \rightarrow \alpha_1.$	

Дійсно, після першого застосування алгорифму до слова $\bar{x}_1 * \bar{x}_2 * \dots * \bar{x}_n$ будемо мати слово $\alpha_1 \bar{x}_1 * \bar{x}_2 * \dots * \bar{x}_n$. Після цього $\text{Sub}_1, \text{Sub}_2, \dots, \text{Sub}_{m-1}, \text{Sub}_{m+1}, \dots, \text{Sub}_n$ викреслюють всі x_i від 1 до $m - 1$ і від $m + 1$ до n включно, залишаючи слово x_m . Подробиці рекомендується розглянути самостійно як вправу.

Теорема доведена.

Нормальний алгорифм називається **замкнутим**, якщо його схема містить у собі формулу підстановки виду $e \rightarrow .q$. При роботі такого алгорифму зупинка можлива лише в тому випадку, коли застосовується заключна формула підстановки. Всякий нормальний алгорифм можна перетворити в замкнений алгорифм шляхом добавлення в кінці схеми нової формули підстановки виду $e \rightarrow .e$. Якщо позначити через A схему одержаного алгорифму, то очевидно, що A замкнений і повністю еквівалентний алгорифму A .

Теорема 3.1.5. Композиція двох нормальних алгорифмів є нормальний алгорифм.

Д о в е д е н н я. Нехай A, B — нормальні алгорифми в алфавіті A . Поставимо у відповідність кожній букві b цього алфавіту букву b^a , яку назовемо **двійником** букви b . Нехай A^a — алфавіт

двійників букв алфавіту A . Виберемо ще які-небудь дві букви: α і β , які не належать $A \cup A^\Delta$. Позначимо через S_A схему алгорифму, яку одержуємо із схеми нормальногого алгорифму A заміною в ній крапки в кожній заключній формулі підстановки буквою α , і позначимо через S_B схему, яка є результатом заміни в схемі алгорифму B всіх букв алфавіту A їх двійниками, всіх крапок — буквами β з подальшою заміною всіх формул підстановки виду $e \rightarrow Q$ і $e \rightarrow .Q$ відповідно формулами підстановки $\alpha \rightarrow \alpha Q$ і $\alpha \rightarrow \alpha \beta Q$. Розглянемо схему (в скороченому вигляді):

$$\left\{ \begin{array}{ll} a\alpha \rightarrow \alpha a & (a \in A) \\ \alpha a \rightarrow \alpha a^\Delta & (a \in A) \\ a^\Delta b \rightarrow a^\Delta b^\Delta & (a, b \in A) \\ a^\Delta \beta \rightarrow \beta a^\Delta & (a \in A) \\ \beta a^\Delta \rightarrow \beta a & (a \in A) \\ ab^\Delta \rightarrow ab & (a, b \in A) \\ \alpha \beta \rightarrow .e \\ S_B \\ S_A . \end{array} \right.$$

Нормальний алгорифм Z над алфавітом A , що визначається цією схемою, такий, що для будь-якого слова p в A справедливе співвідношення $Z(p) \approx B(A(p))$ (див. вправу 9 в кінці параграфа). Одержані нормальний алгорифм називається **композицією алгорифмів A і B** і позначається також символом $B \cdot A$. В загальному випадку запис $A_n \cdot A_{n-1} \cdot \dots \cdot A_1$ буде означати $A_n(A_{n-1}(\dots(A_2 \cdot A_1)))$.

Нехай Δ — деякий нормальний алгорифм в алфавіті A і B — деяке розширення алфавіту A . До схеми алгорифму Δ добавимо зверху всі можливі формули підстановки виду $b \rightarrow b$, де b — довільна буква із $B \setminus A$. Одержана схема визначає деякий нормальний алгорифм Δ_B в алфавіті B , який не застосовний ні до якого слова, що складається з букв алфавіту $B \setminus A$, і такий, що $\Delta_B(p) \approx \Delta(p)$ для будь-якого слова p в A . Алгорифм Δ_B повністю еквівалентний алгорифму Δ відносно алфавіту A і називається **формальним поширенням алгорифму Δ на алфавіт B** .

Нехай дано нормальні алгорифми A і B відповідно в алфавітах A_1 і A_2 . Розглянемо алфавіт $A = A_1 \cup A_2$ і формальні поширення A_A і B_A алгорифмів A і B на алфавіт A . Композиція Z алгорифмів $A_A \cdot B_A$ називається **нормальню композицією** алгорифмів A і B і позначається $B \cdot A$. (Непорозумінь з повторним введенням

символу $B \cdot A$ не виникає, оскільки у випадку, коли $A_1 = A_2$, нормальна композиція алгорифмів B і A збігається з їх композицією.) Z є нормальним алгорифмом над A_1 , причому $Z(p) \approx B(A(p))$ для будь-якого слова p в A_1 і, крім того, Z застосовний тільки до тих слів p в алфавіті A , які відповідають умовам:

- (i) p є слово в алфавіті A_1 ;
- (ii) A застосовний до p ;
- (iii) B застосовний до $A(p)$.

Припустимо, що алфавіт B є розширенням алфавіту A , і нехай p — довільне слово в алфавіті B . Проекцією p^A слова p на алфавіт A називається слово, яке одержуємо із слова p , коли в p стерти всі входження букв із $B \setminus A$. Скорочено записана схема $\{\xi \rightarrow e | \xi \in B \setminus A\}$ задає нормальний алгорифм $B_{B,A}$, такий, що $B_{B,A}(p) = p^A$ для будь-якого слова p в B . Алгорифм $B_{B,A}$ називається **алгорифмом-проекцією**.

Нехай A і C — алфавіти без спільних букв. Покладемо $B = A \cup C$. Тоді скорочено записана схема

$$\{ca \rightarrow ac | a \in A, c \in C\}$$

задає нормальний алгорифм $X_{A,C}$ в алфавіті B , такий, що $X_{A,C}(p) = p^A p^C$ для будь-якого слова p в B .

Якщо A — нормальний алгорифм в алфавіті A і B — розширення A , то нормальний алгорифм B в алфавіті B , який задається схемою алгорифму A , називається **звичайним поширенням алгорифму A** на алфавіт B . Очевидно, що $B(p) \approx A(p)$ для будь-якого слова p в A і для будь-якого слова q в $B \setminus A$. Зауважимо, що звичайне поширення алгорифму A на B відрізняється від формального поширення A на B , оскільки формальне поширення не застосовне до жодного слова, що включає букви із $B \setminus A$.

Теорема 3.1.6. *Нехай A_1, \dots, A_k — нормальні алгорифми і A — об'єднання їх алфавітів. Тоді існує нормальний алгорифм B над A , який називається з'єднанням алгорифмів A_1, \dots, A_k , такий, що $B(p) \approx A_1^{\#}(p) A_2^{\#}(p) \dots A_k^{\#}(p)$ для будь-якого слова p в алфавіті A , де $A_i^{\#}$ — звичайне поширення A_i на A .*

Д о в е д е н и я. Покажемо спочатку, що дана теорема має місце для $k = 2$, а потім індукцією за числом k можна довести і загальний випадок. Введемо алфавіт двійників $A^{\#}$ букв алфавіту A . Покладемо $B = A \cup A^{\#}$. Нехай \bar{A}_1 — нормальний алгорифм, схема якого є результатом заміни кожної букви схеми алгорифму A_1 її двійником, і нехай $\bar{A}_1^{\#}, \bar{A}_2^{\#}$ — звичайні поширення відповідно алгорифмів \bar{A}_1 і \bar{A}_2 на B . Нехай $A = \{a_1, \dots, a_2\}$ і $M =$

$= \text{Sub}(a_1, \dots, a_k : q_1, \dots, q_k)$ — нормальний алгорифм, який задається схемою

$$\begin{cases} \alpha a_i \rightarrow q_i \alpha & (i = 1, 2, \dots, k) \\ \alpha \xi \rightarrow \xi \alpha & (\xi \in A \setminus \{a_1, \dots, a_k\}) \\ \alpha \rightarrow .e \\ e \rightarrow \alpha \end{cases}$$

і виконує одночасну підстановку слів q_1, \dots, q_k в слово p відповідно замість букв a_1, \dots, a_k (див. вправу 9 в кінці параграфа). Тоді нормальні алгорифми $M_1 = \text{Sub}(a_1, \dots, a_n : a_1 a_1^\Delta, \dots, a_n a_n^\Delta)$ і $M_2 = \text{Sub}(a_1^\Delta, \dots, a_n^\Delta : a_1, \dots, a_n)$ над B такі, що M_1 виконує одночасну підстановку $a_1 a_1^\Delta, \dots, a_n a_n^\Delta$ замість a_1, \dots, a_n , а M_2 — одночасну підстановку a_1, \dots, a_n замість $a_1^\Delta, \dots, a_n^\Delta$. Існують, крім того, нормальні алгорифми X_{A, A^Δ} і $X_{A^\Delta, A}$, такі, що $X_{A, A^\Delta}(p) = p^\Delta p^{\Delta^\Delta}$, $X_{A^\Delta, A}(p) = p^{A^\Delta} p^\Delta$. Тоді, як неважко перевірити, нормальнна композиція $B = M_2 \cdot \bar{A}_1^\# \cdot X_{A^\Delta, A} \cdots \bar{A}_2^\# \cdot X_{A, A^\Delta} \cdot M_1$ має шукану властивість: $B(p) = A_1^\#(p) A_2^\#(p)$ для будь-якого слова p в алфавіті A .

Доведення за індукцією загального випадку пропонується як вправа.

Наслідок 3.1.1. Нехай A_1, \dots, A_k — нормальні алгорифми відповідно над алфавітами A_1, \dots, A_k і нехай $A = A_1 \cup \dots \cup A_k$. Тоді існує нормальний алгорифм B над $A \cup \{*\}$, такий, що $B(p) \approx A_1^\#(p) * A_2^\#(p) * \dots * A_k^\#(p)$ для будь-якого слова p в A , де $A_i^\#$ — звичайне поширення на A , і, зокрема, $B(p) \approx A_1(p) * A_2(p) * \dots * A_k(p)$ для будь-якого слова p в алфавіті $A_1 \cap \dots \cap A_k$.

Д о в е д е н н я. Існує такий нормальній алгорифм Δ в $A \cup \{*\}$, що $\Delta(p) = *$ для будь-якого слова p в A . Алгорифм Δ задається схемою

$$\begin{cases} a \rightarrow e & (a \in A) \\ e \rightarrow .* \end{cases}$$

Тоді на основі теореми 3.1.6 $B \in \text{з'єднанням алгорифмів } A_1, \Delta, A_2, \Delta, \dots, \Delta, A_k$. Легко помітити, що $B(p) \approx A_1^\#(p) * A_2^\#(p) * \dots * A_k^\#(p)$ для будь-якого слова p в A і, зокрема, $B(p) \approx A_1(p) * A_2(p) * \dots * A_k(p)$ для будь-якого слова p в алфавіті $A_1 \cap \dots \cap A_k$.

Теорема 3.1.7. 1. Нехай X — нормальній алгорифм в алфавіті A і α — довільна буква. Тоді існує нормальній алгорифм Δ над $A \cup \{\alpha\}$, такий, що для будь-якого слова p в A

$$\Delta(p) = \begin{cases} \alpha p, \text{ якщо } X(p) = e, \\ p, \text{ якщо } X(p) \neq e, \end{cases}$$

i алгорифм Δ застосовний тільки до тих слів, до яких застосовний X .

2. Якщо $A \cup B$ — нормальні алгорифми в алфавіті $A \cup \{\alpha\}$ — буква, що не належить A , то існує нормальний алгорифм P над $A \cup \{\alpha\}$, такий, що $P(p) \approx A(p)$ і $P(\alpha p) \approx B(p)$ для будь-якого слова p в A .

Д о в е д е н н я. 1. Існує нормальний алгорифм B_1 над $A \cup \{\alpha\}$, який переробляє пусте слово e в α і всяке непусте слово в алфавіті $A \cup \{\alpha\}$ в пусте слово e . Такий алгорифм можна задати, наприклад, схемою

$$\begin{cases} a \rightarrow \beta & (a \in A \cup \{\alpha\}), \\ \beta\beta \rightarrow \beta \\ \beta \rightarrow .e \\ e \rightarrow .\alpha \end{cases}$$

де β — буква, яка не належить алфавіту $A \cup \{\alpha\}$.

Нехай $B_2 = B_1 \cdot X$. Для будь-якого слова p в A , якщо $X(p) = e$, то $B_2(p) = \alpha$, і якщо $X(p) \neq e$, то $B_2(p) = e$. Нехай T — тотожний нормальній алгорифм (схема якого має вид $\{e \rightarrow .e\}$), а Δ — з'єднання алгорифмів B_2 і T . Тоді якщо $X(p) = e$, то $\Delta(p) = \alpha p$, і якщо $X(p) \neq e$, то $\Delta(p) = p$.

2. Введемо алфавіт A^Δ двійників букв алфавіту A . Нехай $B = A \cup A^\Delta \cup \{\alpha, \beta\}$, де $\beta \notin A \cup A^\Delta \cup \{\alpha\}$. Якщо замінити в схемі алгорифму B всяку букву алфавіту A її двійником, всі крапки буквою β і в одержаній схемі замінити кожну формулу підстановки вигляду $e \rightarrow q$ і $e \rightarrow .q$ відповідно на $\alpha \rightarrow \alpha q$ і $\alpha \rightarrow \alpha \beta q$, то одержимо деяку схему, яку позначимо $\sum_{\bar{B}}$. Нехай \sum_{A^*} — схема алгорифму A . Побудуємо тепер схему

$$\begin{cases} \alpha a \rightarrow \alpha a^\Delta & (a \in A) \\ a^\Delta b \rightarrow a^\Delta b^\Delta & (a, b \in A) \\ a^\Delta \beta \rightarrow \beta a^\Delta & (a \in A) \\ \beta a^\Delta \rightarrow \beta a & (a \in A) \\ ab^\Delta \rightarrow ab & (a, b \in A) \\ \alpha \beta \rightarrow .e \\ \sum_{\bar{B}} \\ \sum_{A^*} \end{cases}.$$

Нормальний алгорифм Γ , який задається цією схемою над $A \cup \{\alpha\}$, є шуканим алгорифмом, тобто $\Gamma(p) \approx A(p)$ і $\Gamma(\alpha p) \approx B(p)$ для всякого слова p в A .

Теорема 3.1.8. Нехай A, B, X — нормальні алгорифми і A — об'єднання їх алфавітів. Тоді існує нормальний алгорифм Γ над A , такий, що

$$\Gamma(p) = \begin{cases} B(p), & \text{якщо } p - \text{слово в } A \text{ і } X(p) = e, \\ A(p), & \text{якщо } p - \text{слово в } A \text{ і } X(p) \neq e, \end{cases}$$

і який застосовний до тих і тільки тих слів в A , до яких застосовний X . Алгорифм Γ називається **розгалуженням** алгорифмів A і B , яке керується алгорифмом X .

Д о в е д е н н я. Нехай A_1, B_1, X_1 — формальні поширення відповідно алгорифмів A, B, X на A і α — буква, що не належить A . За теоремою 3.1.7(1) існує такий нормальний алгорифм Δ над $A \cup \{\alpha\}$, що

$$\Delta(p) = \begin{cases} \alpha p, & \text{якщо } p - \text{слово в } A \text{ і } \Gamma(p) = e, \\ p, & \text{якщо } p - \text{слово в } A \text{ і } \Gamma(p) \neq e. \end{cases}$$

Крім того, за теоремою 3.1.7(2), існує такий нормальний алгорифм Φ над $A \cup \{\alpha\}$, що коли p — слово в алфавіті A , то $\Phi(p) \approx A_1(p)$ і $\Phi(\alpha p) \approx B_1(p)$. Тепер залишається покласти $\Lambda = \Phi \cdot \Delta$.

Теорема доведена.

Нехай задані алгорифми A і X в алфавіті A і довільне слово p_0 в A . Застосуємо A до p_0 . Якщо одержимо деяке слово p_1 , то застосуємо X до p_1 . Якщо виявиться, що $X(p_1) = e$, то процес закінчується, якщо $X(p_1) \neq e$, то знову застосуємо A до p_1 . Коли внаслідок цього отримаємо деяке слово p_2 , то застосуємо X до p_2 , і знову, якщо виявиться, що $X(p_2) = e$, то процес зупиниться, а якщо $X(p_2) \neq e$, то застосуємо A до p_2 і т.д. Визначений таким чином алгорифм B називається **повторенням** алгорифму A , яке керується алгорифмом X . Очевидно, що $B(p_0) = q$ тоді і тільки тоді, коли існує послідовність слів p_0, p_1, \dots, p_n ($n > 0$), така, що $p_n = q$, $X(p_n) = e$, $p_i = A(p_{i-1})$ при $0 < i \leq n$ і $X(p_i) \neq e$ при $0 < i < n$.

Теорема 3.1.9. Нехай A і X — нормальні алгорифми, A — об'єднання їх алфавітів і A_1 і X_1 — формальні поширення відповідно A і X на A . Тоді існує нормальний алгорифм B над A , який є повторенням алгорифму A_1 , що керується алгорифмом X_1 .

Д о в е д е н н я. Теорему, очевидно, достатньо довести для випадку, коли алфавіти алгорифмів A і X збігаються, і тоді $A_1 = A$ і $X_1 = X$. Нехай буква α не належить A . За теоремою 3.1.7, п. 1 існує такий нормальний алгорифм Δ над $B = A \cup \{\alpha\}$, що

$$\Delta(p) = \begin{cases} \alpha p, & \text{якщо } p - \text{слово в } A \text{ і } \Gamma(p) = e, \\ p, & \text{якщо } p - \text{слово в } A \text{ і } \Gamma(p) \neq e. \end{cases}$$

Нехай $T = \Delta \cdot A$, T — нормальний алгорифм у деякому розширенні F алфавіту B . Нехай буква β не належить F . Розглянемо таку схему:

$$\begin{cases} \xi\beta \rightarrow \beta\xi & (\xi \in F) \\ \beta\alpha \rightarrow .\alpha , \\ \beta \rightarrow e & , \\ \sum_{T^B} & , \end{cases}$$

де \sum_{T^B} — схема, одержана зі схеми T шляхом заміни в ній всіх крапок буквою β . Ця схема задає деякий нормальній алгорифм Φ , такий, що $\Phi(p) = q$ тоді і тільки тоді, коли існує послідовність слів p_0, p_1, \dots, p_n , така, що $p_n = q$, $p_i = T(p_{i-1})$ ($0 < i \leq n$), і p_n єдине в цій послідовності слово, що починається з букви α . Нехай H — алгорифм, який проектує алфавіт F на алфавіт $F \setminus \{\alpha\}$ (тобто алгорифм, що знищує всі входження букви α). Тепер легко переконатися, що нормальній алгорифм $B = H \cdot \Phi$ — шуканий.

Теорема доведена.

Наслідок 3.1.2. Нехай A і B — нормальні алгорифми і A — об'єднання їх алфавітів. Тоді існує нормальний алгорифм Z над A , який всяке слово p в алфавіті A переробляє в слово q тоді і тільки тоді, коли існує така послідовність слів p_0, \dots, p_n ($n \geq 0$), що $p_0 = p$, $p_n = q$, $B(p_n) = e$, $p_{i+1} = A(p_i)$ і $B(p_i) \neq e$ для $i = 0, 1, \dots, n-1$.

Д о в е д е н н я. Нехай T — тотожний нормальній алгорифм і B — повторення алгорифму A , під керуванням Z . Шуканим алгорифмом Δ тоді є розгалуження B і T під керуванням Z (див. теорему 3.1.8). Цей алгорифм Δ називається *повним повторенням* алгорифму A під керуванням Z .

Теорема 3.1.10. Який би не був нормальній алгорифм A в алфавіті A , існує такий нормальній алгорифм A' над алфавітом $B = A \cup C$, де $C = \{1, *\}$, що для будь-якого слова p в A і будь-якого натурального числа k маємо $A'(\bar{k} * p) = Q$ тоді і тільки тоді, коли існує послідовність слів p_0, p_1, \dots, p_k ($k \geq 0$), яка задоволяє умови: $p_0 = p$, $p_k = Q$, $p_i = A(p_{i-1})$, для $i = 1, 2, \dots, n$.

Д о в е д е н н я. Нехай α — яка-небудь буква, що не входить до алфавіту B . Покладемо $D = B \cup \{\alpha\}$. Розглянемо нормальні алгорифми в D , які задаються такими схемами:

$$B_1 = \begin{cases} \alpha 11 \rightarrow .1 \\ \alpha 1* \rightarrow \alpha * \\ \alpha * \xi \rightarrow \alpha * & (\xi \in B) \\ \alpha * \rightarrow .e \\ e \rightarrow \alpha \end{cases}$$

Неважко переконатися, що $B_1(\bar{0} * p) = e$ і $B_1(\bar{n} * p) \neq e$, якщо $n > 0$.

$$B_2 = \begin{cases} * \xi \rightarrow * & (\xi \in B), \\ * \rightarrow e. \end{cases}$$

Якщо p не включає символа $*$, то $B_2(p * Q) = p$.

$$B_3 = \begin{cases} \alpha \cdot 1 \rightarrow \alpha \\ \alpha \cdot * \rightarrow .e \\ e \rightarrow \alpha. \end{cases}$$

Очевидно, що $B_3(\bar{n} * p) = p$ для будь-якого $p \in F(B)$.

$$B_4 = \{1 \rightarrow .e\}$$

$$B_5 = \{1^* \rightarrow .e\}$$

Очевидно, що $B_4(\bar{n} * p) = \overline{n-1} * p$, коли $n > 0$, і $B_4(\bar{0} * p) = *p$. Крім того, $B_5(\bar{0} * p) = p$.

Нехай тепер Z — такий нормальній алгорифм, що $Z(p) = (B_2 \cdot B_4)(p) * (A \cdot B_3)(p)$ для будь-якого слова $p \in F(D)$ (див. наслідок 3.1.1). Звідси одержуємо, що $\forall p \in F(A)$

$$Z(\bar{n} * p) = \begin{cases} \overline{n-1} * A(p), & \text{якщо } n > 0, \\ * B(p), & \text{якщо } n = 0, \end{cases}$$

де B — повне повторення алгорифму A . Позначимо через E алфавіт алгорифму Z . Із 3.1.2 випливає, що знайдеться такий нормальній алгорифм Φ над E , що $\Phi(p) = Q$ тоді і тільки тоді, коли існує послідовність слів p_0, p_1, \dots, p_k ($k \geq 0$), яка задовільняє умови:

$$\begin{aligned} p_0 &= p, p_k = Q, B_1(p_k) = e, p_i = Z(p_{i-1}) \quad (0 < i \leq k) \\ &\text{i } B(p_i) \neq e \quad (0 \leq i < k). \end{aligned}$$

Тепер неважко переконатися, що $A' = B_5 \cdot \Phi$.

Теорема доведена.

Теорема 3.1.11. Всяка частково рекурсивна функція частково обчислювана за Марковим.

Д о в е д е н и я. Для того щоб довести теорему, необхідно показати обчислюваність за Марковим найпростіших функцій, а також обчислюваність операцій суперпозиції, примітивної рекурсії і мінімізації частково рекурсивних функцій.

Обчислюваність за Марковим найпростіших функцій o, s і I_n^m випливає з теореми 3.1.4.

Операція суперпозиції. Нехай функція ϕ побудована за функціями f, f_1, \dots, f_k за допомогою операції суперпозиції:

$$\varphi(x_1, \dots, x_n) = f(f_1(x_1, \dots, x_n), \dots, f_k(x_1, \dots, x_n)),$$

де f, f_1, \dots, f_k — частково рекурсивні функції. Припустимо, що існують нормальні алгорифми $A_f, A_{f_1}, \dots, A_{f_k}$ над $M = \{1, *\}$, які частково обчислюють відповідні функції. Тоді за наслідком 3.1.1 існує нормальний алгорифм B над M , такий, що

$$B(p) \approx A_{f_1}(p) * A_{f_2}(p) * \dots * A_{f_k}(p)$$

для будь-якого $p \in F(M)$. Зокрема,

$$\overline{B(x_1, \dots, x_n)} = \overline{f_1(x_1, \dots, x_n)} * \dots * \overline{f_k(x_1, \dots, x_n)}$$

для будь-яких натуральних чисел x_1, x_2, \dots, x_n . Покладемо $Z = A_f \cdot B$. Тоді для будь-яких натуральних чисел x_1, x_2, \dots, x_n будемо мати

$$\begin{aligned} Z((\overline{x_1, \dots, x_n})) &= A_\varphi(\overline{f_1(x_1, \dots, x_n)}, \dots, \overline{f_k(x_1, \dots, x_n)}) = \\ &= \varphi(\overline{f_1(x_1, \dots, x_n)}, \dots, \overline{f_k(x_1, \dots, x_n)}). \end{aligned}$$

Операція рекурсії. Нехай функція f побудована за функціями g і h за допомогою операції примітивної рекурсії

$$\begin{aligned} f(x_1, \dots, x_n, 0) &= g(x_1, \dots, x_n), \\ f(x_1, \dots, x_n, y+1) &= h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))), \end{aligned}$$

і нехай A_g і A_h — нормальні алгорифми над M , які частково обчислюють відповідно функції g і h . Нарешті, нехай A_0, A_s і A_m^n — нормальні алгорифми, які обчислюють найпростіші функції o, s і I_m^n . За допомогою алгорифмів A_m^{n+1} можна побудувати (за наслідком 3.1.1) такий нормальній алгорифм B_1 над M , що $B_1(\overline{x_1}, \dots, \overline{x_n}, \overline{y}) = \overline{x_1} * \dots * \overline{x_n}$.

Нехай $B = A_g \cdot B_1$. Виходячи з алгорифмів $A_{k+1}^{k+1}, A_1^{k+1}, \dots, A_k^{k+1}, A_0, B$ і наслідку 3.1.1, будуємо нормальній алгорифм B_2 над M , такий, що

$$B_2(\overline{x_1} * \dots * \overline{x_n} * \overline{y}) \approx \overline{y} * \overline{x_1} * \dots * \overline{x_n} * \overline{0} * \overline{g(x_1, \dots, x_n)}.$$

Нормальний алгорифм $B_3 = A_s \cdot A_{k+1}^{k+2}$ працює так, що $B_3(\overline{x_1} * \dots * \overline{x_n} * \overline{y} * \overline{x}) = \overline{y+1}$. Знову, застосовуючи наслідок 3.1.1 до алгорифмів $A_1^{k+2}, \dots, A_k^{k+2}, B_3$ і A_f , одержуємо нормальній алгорифм B_4 над M , такий, що

$$B_4(\overline{x_1} * \dots * \overline{x_n} * \overline{y} * \overline{x}) = \overline{x_1} * \dots * \overline{x_n} * \overline{y+1} * \overline{h(x_1, \dots, x_n, y, x)}.$$

За теоремою 3.1.10 існує такий нормальній алгорифм B'_4 , що при будь-якому $n \geq 0$ рівність $B'_4(\bar{n} * p) = Q$ виконується тоді і тільки тоді, коли існує послідовність слів p_0, p_1, \dots, p_n , яка задовільняє умови $p_0 = p$, $p_n = Q$ і $p_i = B_4(p_{i-1})$, де $0 < i \leq n$. Тоді алгорифм $B = A_{k+2}^{k+2} \cdot B'_4 \cdot B_2$ і є шуканим алгорифмом, який обчислює функцію f . Дійсно,

$$B_2(\bar{x}_1 * \dots * \bar{x}_n * \bar{y}) = \bar{y} * \bar{x}_1 * \dots * \bar{x}_n * \bar{0} * \overline{g(x_1, \dots, x_n)}.$$

Застосування B'_4 до слова $\bar{y} * \bar{x}_1 * \dots * \bar{x}_n * \bar{0} * \overline{g(x_1, \dots, x_n)}$, очевидно, рівносильне y -кратному застосуванню B_4 , починаючи зі слова $\bar{x}_1 * \dots * \bar{x}_n * \bar{0} * \overline{g(x_1, \dots, x_n)}$. Неважко зрозуміти, що при цьому одержуємо слово $\bar{x}_1 * \dots * \bar{x}_n * \bar{y} * \overline{f(x_1 * \dots * x_n * y)}$. Застосувавши до цього слова алгорифм A_{k+2}^{k+2} , одержуємо $\overline{f(x_1 * \dots * x_n * y)}$.

Операція мінімізації. Нехай $f(x_1, \dots, x_n) = \mu_y(g(x_1, \dots, x_n, y) = 0)$ і A_g — нормальний алгорифм над M , який обчислює функцію g . Виходячи з алгорифмів $A_1^{n+1}, \dots, A_n^{n+1}$ і $A_s A_n^{n+1}$, побудуємо алгорифм M , такий, що $M(\bar{x}_1 * \dots * \bar{x}_n * \bar{y}) = \bar{x}_1 * \dots * \bar{x}_n * \overline{y + 1}$. Розглянемо нормальній алгорифм Δ над M , який визначається схемою

$$\begin{cases} 11 \rightarrow .1 \\ 1 \rightarrow e. \end{cases}.$$

Якщо $n = 0$, то $\Delta(n) = e$, а якщо $n > 0$, то $\Delta(n) \neq e$. Нехай $Z = \Delta \cdot A_g$, тоді

$$Z(\bar{x}_1 * \dots * \bar{x}_n * \bar{y}) \begin{cases} = e, g(x_1, \dots, x_n, y) = 0, \\ \neq e, g(x_1, \dots, x_n, y) \neq 0. \end{cases}$$

Нехай тепер P — нормальній алгорифм над M такий, що $P(\bar{x}_1 * \dots * \bar{x}_n * \bar{y}) = \bar{x}_1 * \dots * \bar{x}_n * \bar{0}$. З наслідку 3.1.2 випливає, що для алгорифмів M і Z існує нормальній алгорифм B над M , для якого рівність $B(p) = Q$ виконується тоді і тільки тоді, коли існує послідовність p_0, p_1, \dots, p_n , така, що $p_0 = p$, $p_n = Q$, $Z(p_n) = e$, $p_{i+1} = M(p_i)$ і $Z(p_i) \neq e$ для $i = 0, \dots, n - 1$. Визначимо тепер нормальній алгорифм $\Gamma = A_{n+1}^{n+1} \cdot B \cdot P$. Неважко бачити, що

$$\Gamma(\bar{x}_1 * \dots * \bar{x}_n) = \overline{\mu_y(g(x_1, \dots, x_n, y) = 0)} = \overline{f(x_1, \dots, x_n)}.$$

Отже, якщо $f(x_1, \dots, x_n)$ — частково рекурсивна функція, то існує деякий нормальній алгорифм, такий, що

$$A_f(\bar{x}_1 * \dots * \bar{x}_n) = \overline{f(x_1, \dots, x_n)}.$$

Нехай

$$A(\bar{x}_1 * \dots * \bar{x}_n) = A''(\bar{x}_1 * \dots * \bar{x}_n) * \dots * A''_n(\bar{x}_1 * \dots * \bar{x}_n) = \bar{x}_1 * \dots * \bar{x}_n.$$

Тоді

$$Z(\bar{x}_1 * \dots * \bar{x}_n) = A_f \cdot A(\bar{x}_1 * \dots * \bar{x}_n) = \overline{f(x_1, \dots, x_n)},$$

причому Z застосовний до тих і тільки тих слів p із $F(M)$, які мають вигляд $\bar{x}_1 * \dots * \bar{x}_n$ і для яких $f(x_1, \dots, x_n)$ визначена.

Теорема доведена.

Має місце і обернена теорема.

Теорема 3.1.12. *Всяка частково обчислювана за Марковим функцією є частково рекурсивною.*

Доведення цієї теореми виходить за межі матеріалу даної книги і тому не наводиться. При необхідності це доведення можна знайти, наприклад, у монографії [53].

З теорем 3.1.10 і 3.1.11 випливає, що все, що обчислюється, можна обчислити у вільних моноїдах. Цей факт виявляє важливість поняття вільного моноїда і його роль у теорії алгорифмів. У наступних розділах ми ще повернемось до понять нормального алгорифму Маркова і вільного моноїда.

Контрольні питання

1. Які функції називаються найпростішими?
2. Дайте визначення операцій суперпозиції, примітивної рекурсії і мінімізації.
3. Яка функція називається примітивно рекурсивною, частково рекурсивною?
4. Сформулюйте тезу Черча, тезу Тьюрінга.
5. Які команди виконуються в алгоритмічній системі Поста?
6. Чим відрізняється алгоритмічна система Поста від системи Тьюрінга?
7. Що таке формула підстановки?
8. Що собою являє алгорифмічна система Маркова?

Задачі і вправи

1. Доведіть, що:
 - a) $x \div y = s(x) + s(y);$
 - б) $x + (y + x) = y + (x + y);$
 - в) $x \div (y + z) = (x \div y) \div z;$

г) $(x + y) + z = (x + z) + y$.

2. Доведіть, що коли функція $f(x_1, x_2, \dots, x_n)$ примітивно рекурсивна, то наведені нижче функції теж примітивно рекурсивні:

а) $g(x_1, x_2, \dots, x_n) = f(x_2, x_1, x_3, \dots, x_n);$

б) $g(x_1, x_2, \dots, x_n) = f(x_2, x_3, \dots, x_n, x_1);$

в) $g(x_1, x_2, \dots, x_{n+1}) = f(x_1, x_2, \dots, x_n)$ (введення фіктивного аргумента);

г) $g(x_1, x_2, \dots, x_{n-1}) = f(x_1, x_1, x_2, \dots, x_{n-1})$ (ототожнення аргументів).

3. Доведіть, що функція $f(x) = x!$ примітивно рекурсивна.

4. Які функції можна побудувати з функцій f і h за допомогою операції примітивної рекурсії:

а) $f(x) = x$, $h(x, y, z) = z^x;$

б) $f(x) = x$, $h(x, y, z) = x^{yz}?$

5. Доведіть, що наведені нижче функції примітивно рекурсивні:

а) $sg(x) = \begin{cases} 0, & \text{якщо } x = 0, \\ 1, & \text{якщо } x > 0; \end{cases}$

б) $\bar{sg}(x) = \begin{cases} 1, & \text{якщо } x = 0, \\ 0, & \text{якщо } x > 0; \end{cases}$

в) $\max(x, y);$

г) $\min(x, y).$

6. Доведіть, що наведені нижче функції частково рекурсивні:

а) ніде не визначена функція ε ;

б) $f(x, y) = \begin{cases} x - y, & \text{якщо } x \geq y; \\ \text{не визначена в решті випадків}; & \end{cases}$

в) $f(x, y) = \begin{cases} x/y, & \text{якщо } y \text{ ділить } x; \\ \text{не визначена в решті випадків}; & \end{cases}$

г) $f(x, y) = \begin{cases} z, & \text{якщо } z^y = x; \\ \text{не визначена в решті випадків}; & \end{cases}$

д) функція визначена лише в скінченному числі точок.

7. Доведіть, що коли z, g, h — частково рекурсивні функції, то частково рекурсивними будуть і такі функції:

а) $\mu_y [z(x_1, x_2, \dots, x_n, y) = g(x_1, x_2, \dots, x_n, y)];$

б) $\mu_y [z(x_1, x_2, \dots, x_n, y) \neq g(x_1, x_2, \dots, x_n, y)];$

в) $\mu_y [z(x_1, x_2, \dots, x_n, y) \leq g(x_1, x_2, \dots, x_n, y)];$

г) $\mu_y [z(x_1, x_2, \dots, x_n, y) < g(x_1, x_2, \dots, x_n, y)];$

д) $\mu_y [z(x_1, x_2, \dots, x_n, y) = 0 \& g(x_1, x_2, \dots, x_n, y) = 0];$

е) $\mu_y [z(x_1, x_2, \dots, x_n, y) = 0 \vee g(x_1, x_2, \dots, x_n, y) = 0];$

ж) $\mu_y [z(x_1, x_2, \dots, x_n, y) = 0 \vee g(x_1, x_2, \dots, x_n, y) \leq h(x_1, x_2, \dots, x_n, y)].$

8. Покажіть, що алгорифм, який побудовано в теоремі 3.1.5, дійсно є суперпозицією алгорифмів A і B .

Покажіть, що алгорифм, наведений в теоремі 3.1.6, виконує одночасну підстановку слів p_1, \dots, p_k замість слів q_1, \dots, q_k в слові p .

9. Нехай $X = \{a_1, a_2, \dots, a_n\}$ і $p \in F(X)$ — довільне слово. Опишіть дії нормальних алгорифмів Маркова, які задаються схемами:

a) $S = \{ e \rightarrow .p;$

b) $S = \begin{cases} \alpha\xi \rightarrow \xi\alpha \\ \alpha \rightarrow .p \\ e \rightarrow \alpha, \text{ де } \alpha \in X, \xi \notin X; \end{cases}$

b) $S = \begin{cases} a_1 \rightarrow e \\ a_2 \rightarrow e \\ \dots \\ a_n \rightarrow e \\ e \rightarrow .p \end{cases}$

10. Побудуйте нормальні алгорифми, які обчислюють значення операцій `conc`, `head` і `tail` в алфавіті $X = \{a_1, a_2, \dots, a_n\}$.

11. Побудуйте нормальні алгорифми, які обчислюють значення операцій додавання, модуля різниці двох чисел, зрізаної різниці, множення.

12. Покажіть, що наведені нижче функції обчислюються машинами Тьюрінга:

a) $o(n)$ і $I_m^n(k_1, k_n);$

б) $m + n$ і $m \div 1;$

в) $m \div n$ і $m \cdot n;$

г) $\text{conv}(p) = x_kx_{k-1}\dots x_1$, де $p = x_1x_2\dots x_k$.

13. Напишіть програму роботи машини Поста для обчислення значень таких функцій:

а) $o(x) = 0;$

б) $I_k^n(m_1, \dots, m_n) = m_k;$

в) $t(n) = 2n;$

г) $m + n;$

д) $\text{conv}(p) = x_kx_{k-1}\dots x_1$, де $p = x_1x_2\dots x_k$.

§ 3.2. ЧИСЛЕННЯ ВИСЛОВЛЮВАНЬ

1. АЛФАВІТ І ФОРМУЛИ

Нехай $Al = \{A, B, C, \dots, X, Y, A_1, \dots\}$ — алфавіт змінних (скінчений або злічений). Кожна із змінних цього алфавіту може приймати одне із двох значень **ТАК** (істина — 1) або **НІ** (хибність — 0), а самі змінні в цьому випадку називають **висловлюваннями**. Інакше кажучи, висловлювання — це або хибне, або істинне висловлювання, але не те і друге разом.

Приклади 3.2.1 (висловлювання)

A — Сніг білий.

A_1 — Галина — студентка університету.

X — Група САПР налічує 30 студентів.

Елементи алфавіту Al , які використовуються для висловлювань, будемо називати *пропозиційними змінними*, або *атомарними формулами*, або просто *атомами*.

Із висловлювань можна будувати складені висловлювання, використовуючи атомарні формули і логічні зв'язки.

У логіці першого порядку і, зокрема, в логіці висловлювань будемо використовувати п'ять логічних зв'язок:

\neg (ні), $\&$ (і), \vee (або), \rightarrow (якщо..., то), \Leftrightarrow (тоді і тільки тоді).

Приклади 3.2.2 (складені висловлювання)

Сніг білий, і температура нижче нуля.

Якщо Петро на заняттях, то Галина вдома.

Петро — студент інституту або робітник ДОК.

Означення 3.2.1. Правильно побудованими формулами (пpf) в логіці висловлювань являються такі формули.

1. Атомарна формула є пpf.
2. Якщо A є пpf, то $\neg A$ — теж пpf.
3. Якщо G і H — пpf, то $(G \& H)$, $(G \vee H)$, $(G \rightarrow H)$, $(G \Leftrightarrow H)$ — теж пpf.
4. Ніякі інші формули, крім формул, що породжені правилами 1—3, не є пpf.

Пpf ще називають словами в алфавіті Al , побудованими за допомогою логічних зв'язок.

Підформулою пpf A називається всяке підслово слова A , яке саме є пpf.

При записах формул можна обйтись без деяких дужок, якщо приписати ранги логічним зв'язкам. Ранги логічним зв'язкам найчастіше задаються в такому порядку (від найнижчого до найвищого): \Leftrightarrow , \rightarrow , \vee , $\&$, \neg .

Приклади 3.2.3 (висловлювання і формули)

1. P — вологість велика.

2. Q — температура висока.

3. C — ми почуваемо себе добре.

Висловлювання “якщо вологість велика і температура висока, то ми почуваемо себе добре” можна записати у вигляді фор-

мули $((P \& Q) \rightarrow (\neg C))$ або з урахуванням рангів операцій: $P \& Q \rightarrow \neg C$. *

Обчислення значень формул

Значення формул $\neg A$, $A \vee B$, $A \& B$, $A \rightarrow B$ і $A \Leftrightarrow B$ задаються за допомогою таблиць, які називаються **таблицями істинності**.

A	$\neg A$	A	B	$A \vee B$	$A \& B$	$A \rightarrow B$	$A \Leftrightarrow B$
1	0	1	1	1	1	1	1
0	1	1	0	1	0	0	0
		0	1	1	0	1	0
		0	0	0	0	1	1

З цих таблиць видно, що логічні зв'язки, якщо їх розглядати як операції на множині формул, задовольняють закони комутативності, асоціативності, дистрибутивності та інші закони (див. вправу 1 в кінці параграфа 3.3).

2. ІНТЕРПРЕТАЦІЯ ФОРМУЛ ЛОГІКИ ВИСЛОВЛЮВАНЬ

Приписування значень 1 або 0 атомарним формулам, які входять в складені формули, називається *інтерпретацією*. Формально це можна сформулювати так.

Означення 3.2.2. Відображення $h: Al \rightarrow \{0, 1\}$ називається інтерпретацією, якщо для всяких формул A, B числення висловлювань h задовольняє такі умови:

$$\begin{aligned} h(\neg A) &= \neg h(A), \\ h(A \& B) &= h(A) \& h(B), \\ h(A \vee B) &= h(A) \vee h(B). \end{aligned}$$

Отже, задаючи інтерпретацію h , можна обчислити за допомогою таблиць істинності значення довільної заданої формули A .

Приклад 3.2.4

Нехай $A = ((A_1 \& A_2) \vee \neg A_1) \& \neg A_2$ і $h(A_1) = h(A_2) = 0$. Тоді $h(A) = h((A_1 \& A_2) \vee \neg A_1) \& h(\neg A_2) = (h(A_1) \& h(A_2) \vee \neg h(A_1)) \& \neg h(A_2) = ((0 \& 0) \vee \neg 0) \& \neg 0 = (0 \vee 1) \& 1 = 1 \& 1 = 1$. *

Означення 3.2.3. Говорять, що формула A числення висловлювань істинна при деякій інтерпретації h тоді і тільки тоді, коли $h(A) = 1$. В протилежному випадку говорять, що A хибна при інтерпретації h . Якщо $h(A) = 1$, то інтерпретацію h називають моделлю для формули A .

Означення 3.2.4. Формула A числення висловлювань називається *тавтологією (суперечністю)*, якщо вона приймає значення 1 (0) незалежно від інтерпретації.

Означення 3.2.5. Якщо формула $A \rightarrow B$ — тавтологія, то говорять, що із формули A логічно витікає формула B , або що формула B — логічний (семантичний) наслідок формули A в численні висловлювань. Якщо формула $A \Leftrightarrow B$ — тавтологія, то кажуть, що формули A і B логічно (семантично) еквівалентні в численні висловлювань.

Таблиці істинності дають нам ефективну процедуру для вирішення питання: чи є дана формула тавтологією.

Приклади 3.2.5

- Чи є формула $((A \rightarrow B) \rightarrow B) \rightarrow B$ тавтологією?

Розв'язок.

A	B	$A \rightarrow B$	$(A \rightarrow B) \rightarrow B$	$((A \rightarrow B) \rightarrow B) \rightarrow B$
1	0	0	1	0
1	1	1	1	1
0	1	1	1	1
0	0	1	0	1

Відповідь: наведена формула не є тавтологією.

- Відомо, що A і $A \rightarrow B$ — тавтології. Довести, що B — теж тавтологія.

Розв'язок.

A	B	$A \rightarrow B$	Якби B набирала значення 0, то $A \rightarrow B$ була б суперечністю, а це не може бути за умовою.
1	1	1	Отже, B повинна бути теж тавтологією.
1	0	1	

- Чи буде тавтологією формула $A \rightarrow A$?

Розв'язок.

A	$A \rightarrow A$	
1	1	
0	1	Значить, наведена формула є тавтологією. ▲

3. ПОВНІ СИСТЕМИ ЗВ'ЯЗОК

Всяка формула від n змінних породжує відповідну функцію істинності від n аргументів, тобто може розглядатися як булева функція. Логічно еквівалентні формулі породжують одну і ту саму функцію істинності, і виникає питання: а чи всі функції істинності породжуються таким чином?

Теорема 3.2.1. Всяка функція істинності породжується деякою формулою, яка має лише зв'язки \neg , $\&$, \vee .

Доведення. Нехай $f(x_1, x_2, \dots, x_n)$ — задана функція істинності. Очевидно, що f може бути представлена таблицею істинності з 2^n рядками, де кожний рядок містить деякий розподіл значень істинності для змінних x_1, x_2, \dots, x_n і відповідні значення $f(x_1, x_2, \dots, x_n)$. Перенумеруємо рядки цієї таблиці натуральними числами $1, 2, \dots, 2^n$. Нехай для кожного $i = 1, 2, \dots, 2^n$ C_i означає кон'юнкцію $U_1^i \& U_2^i \& \dots \& U_n^i$, де $U_j^i \in A_j$, якщо в i -му рядку таблиці істинності x_j набирає значення 1, або $U_j^i \in \neg A_j$, якщо x_j набирає значення 0. Нехай D — диз'юнкція всіх C_i , таких, що f в i -му рядку набирає значення 1.

Якщо таких рядків немає, то f завжди 0, і тоді нашу теорему задовольняє формула $(A_1 \& \neg A_1)$. Покажемо, що таблиця істинності D збігається з f . Дійсно, нехай дано деякий розподіл значень істинності атомарних формул A_1, A_2, \dots, A_n , і відповідне значення f для цього рядка представлено k -м рядком. Тоді C_k має значення 1, в той час як інші C_i мають значення 0.

Якщо значення C_k дорівнює 1, то $C_k \in D$, і тоді D набирає значення 1. Якщо ж f має значення 0 на цьому рядку, то C_k не входить в D , і оскільки решта C_i хибні на цьому наборі, то і D хибна на цьому ж наборі, що і треба було довести.

Приклади 3.2.6

1.	x_1	x_2	$f(x_1, x_2)$	
	1	1	0	
	0	1	1	$C_1 = \neg A_1 \& A_2,$
	1	0	1	$C_2 = A_1 \& \neg A_2,$
	0	0	1	$C_3 = \neg A_1 \& \neg A_2.$

Отже, $D = (\neg A_1 \& A_2) \vee (A_1 \& \neg A_2) \vee (\neg A_1 \& \neg A_2)$.

2.	x_1	x_2	$f(x_1, x_2)$	
	1	1	1	$C_1 = A_1 \& A_2,$
	0	1	1	$C_2 = \neg A_1 \& A_2,$
	1	0	0	
	0	0	1	$C_3 = \neg A_1 \& \neg A_2.$

Таким чином, $D = (A_1 \& A_2) \vee (\neg A_1 \& A_2) \vee (\neg A_1 \& \neg A_2)$. Перетворюючи дану формулу за законами асоціативності, комутативності і дистрибутивності, як зазначалось вище, маємо:

$$D = (\neg A_1 \& \neg A_2) \vee ((A_1 \vee \neg A_1) \& A_2) = (\neg A_1 \& \neg A_2) \vee A_2 = (\neg A_1 \vee A_2) \& (\neg A_2 \vee A_2) = \neg A_1 \vee A_2.$$

Зауважимо, що $f(x_1, x_2)$ якраз і є функцією істинності для формулі $A_1 \rightarrow A_2$, тобто

$$A_1 \rightarrow A_2 \Leftrightarrow \neg A_1 \vee A_2.$$

3.	x_1	x_2	$f(x_1, x_2)$	
	1	1	1	$C_1 = A_1 \& A_2 ,$
	0	1	0	
	1	0	0	
	0	0	1	$C_2 = \neg A_1 \& \neg A_2 .$

Значить, $D = (\neg A_1 \& \neg A_2) \vee (A_1 \& A_2) = (\neg A_1 \vee (A_1 \& A_2)) \& \& (\neg A_2 \vee (A_1 \& A_2)) = ((\neg A_1 \vee A_1) \& (\neg A_1 \vee A_2)) \& ((\neg A_2 \vee A_1) \& \& (\neg A_2 \vee A_2)) = (\neg A_1 \vee A_2) \& (\neg A_2 \vee A_1).$

Таким чином,

$$A_1 \Leftrightarrow A_2 = (\neg A_1 \vee A_2) \& (\neg A_2 \vee A_1) = (A_1 \rightarrow A_2) \& (A_2 \rightarrow A_1). \quad \blacktriangleleft$$

Неважко перевірити, що мають місце такі формулі:

$$\begin{aligned} A \& B \Leftrightarrow \neg(\neg A \vee \neg B) &\Leftrightarrow \neg(A \rightarrow \neg B), \\ A \vee B \Leftrightarrow \neg(\neg A \& \neg B) &\Leftrightarrow \neg A \rightarrow B. \end{aligned}$$

Теорема 3.2.2. Всяка функція істинності f породжується формuloю, яка має лише одну з таких пар логічних зв'язок: $(\neg, \&)$, (\neg, \vee) , (\rightarrow, \neg) .

Теорема 3.2.2 являє собою узагальнення теореми про повноту системи відповідних операцій для булевої алгебри.

4. СИСТЕМА АКСІОМ ДЛЯ ЧИСЛЕННЯ ВИСЛОВЛЮВАНЬ

Формальна аксіоматична теорія Th вважається визначеною, коли виконані такі умови.

1. Задано деякий злічений алфавіт — символи теорії Th . Скінченні послідовності символів теорії Th називаються *виразами Th*.

2. Задано підмножину виразів теорії Th , яка називається *множиною формул теорії Th*. (Часто існує процедура, за допомогою якої можна завжди визначити, чи є даний вираз формuloю.)

3. Задано підмножину множини формул, елементи якої називаються *аксіомами теорії Th*. (Якщо є можливість перевірити, чи є дана формула аксіомою, то Th називається *ефективно аксіоматизованою або аксіоматичною теорією*.)

4. Задано скінченну множину R_1, R_2, \dots, R_n відношень між формулами, які називаються *правилами виведення*. Для будь-якого відношення $R_i \exists j$ із N^+ , такого, що для множини формул

A_1, A_2, \dots, A_j і всякої формулі A ефективно вирішується питання про те, чи знаходяться дані формулі A_1, A_2, \dots, A_j у відношенні R_i з формuloю A , чи ні. Коли $(A_1, A_2, \dots, A_j, A) \in R_i$, то A називається **безпосереднім наслідком** формул A_1, A_2, \dots, A_j за правилом $R_i(A_1, \dots, A_j) \vdash A$.

Виведенням у \mathcal{Th} називається всяка послідовність формул A_1, A_2, \dots, A_n , така, що будь-яка з формул A_i є або аксіомою, або безпосереднім наслідком деяких попередніх формул за одним із правил виведення.

Формула A називається **теоремою теорії \mathcal{Th}** , коли існує виведення в \mathcal{Th} формулі A , в якому останнім елементом є формула A . Це виведення називається **виведенням формули A в \mathcal{Th}** .

Якщо є алгоритм для перевірки, чи A є теоремою теорії \mathcal{Th} , то теорія \mathcal{Th} називається **розв'язною** теорією, інакше — теорією, що **не є розв'язною**.

Формула A називається **наслідком у \mathcal{Th}** множини формул Γ тоді і тільки тоді, коли існує така послідовність формул A_1, \dots, A_n , що $A_n = A$, і будь-яка з формул A_i є або аксіомою, або елементом Γ , або безпосереднім наслідком деяких попередніх формул за одним із правил виведення.

Послідовність A_1, \dots, A_n називається **виведенням A із Γ** , а елементи множини Γ — **гіпотезами виведення**.

Запис $\Gamma \vdash A$ означає, що A є наслідком множини формул Γ .

Формальна аксіоматична теорія \mathcal{Th} для числення висловлювань задається таким способом.

1. Символами \mathcal{Th} є $\neg, \rightarrow, (,$) і букви A_i з алфавіту Al . Символи \neg, \rightarrow називаються **примітивними зв'язками**, а A_i — **пропозиційними змінними** або **пропозиційними буквами**.

2. Всі A_i є формулами \mathcal{Th} . Якщо A і B — формулі \mathcal{Th} , то $(\neg A)$, $(A \rightarrow B)$ — формулі \mathcal{Th} .

3. Для будь-яких формул A, B, C теорії \mathcal{Th} формули

$$A1) A \rightarrow (B \rightarrow A);$$

$$A2) (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C));$$

$$A3) (\neg B \rightarrow \neg A) \rightarrow ((\neg B \rightarrow A) \rightarrow B)$$

є аксіомами (точніше, схемами аксіом).

4. Єдиним правилом виведення служить правило ***modus ponens*** (MP): із формул A і $A \rightarrow B$ виводиться B , скорочено:

$$A, A \rightarrow B \vdash B.$$

Інші зв'язки можна ввести за допомогою вже відомих нам формул:

$$A \& B \Leftrightarrow \neg(A \rightarrow \neg B),$$

$$A \vee B \Leftrightarrow \neg A \rightarrow B,$$

$$A \Leftrightarrow B \Leftrightarrow (A \rightarrow B) \ \& \ (B \rightarrow A).$$

Приклад 3.2.7

1. $A \rightarrow A \in \text{Th}.$

Доведення. Підставимо $A \rightarrow A$ в аксіому А2. $(A \rightarrow ((A \rightarrow A) \rightarrow A)) \rightarrow ((A \rightarrow (A \rightarrow A)) \rightarrow (A \rightarrow A))$. Але $(A \rightarrow ((A \rightarrow A) \rightarrow A))$ — це аксіома А1, і за правилом MP одержуємо $(A \rightarrow (A \rightarrow A)) \rightarrow (A \rightarrow A)$. Але $A \rightarrow (A \rightarrow A)$ — це аксіома А1, і знову за правилом MP одержуємо $A \rightarrow A$.

2. $((\neg A \rightarrow A) \rightarrow A) \in \text{Th}.$

Доведення. За А3 маємо $(\neg A \rightarrow \neg A) \rightarrow ((\neg A \rightarrow A) \rightarrow A)$. За попередньою теоремою і за правилом MP маємо $(\neg A \rightarrow A) \rightarrow A$. \blacktriangleleft

Якщо X — деяка підмножина множини аксіом Y , то множина X називається **незалежною**, коли жодна формула з множини X не може бути виведена за допомогою правил виведення із множини аксіом $Y \setminus X$.

Має місце така теорема.

Теорема 3.2.3. *Кожна з аксіом А1—А3 незалежна.*

Доведення цієї теореми опускається, але при необхідності його можна знайти, наприклад, в [32, 54].

Система аксіом А1 — А3 теорії Th не єдина.

Існують і інші аксіоматизації числення висловлювань. Наведемо деякі з них.

Аксіоматизація Гільберта—Аккермана. Логічними зв'язками є \neg і \vee , а $A \rightarrow B$ служить скороченням для формули $\neg A \vee B$.

Аксіоми: ГА1) $A \vee A \rightarrow A$;

ГА2) $A \rightarrow A \vee B$;

ГА3) $A \vee B \rightarrow B \vee A$;

ГА4) $(B \rightarrow C) \rightarrow ((A \vee B) \rightarrow (A \vee C))$.

Правило виведення: MP.

Аксіоматизація Россера. Логічні зв'язки: $\&$, \neg , а $A \rightarrow B$ — скорочення для формули $\neg(A \& \neg B)$.

Аксіоми: РА1) $A \rightarrow (A \& A)$;

РА2) $(A \& B) \rightarrow A$;

РА3) $(A \rightarrow B) \rightarrow (\neg(B \& C) \rightarrow \neg(C \& A))$.

Правило виведення: MP.

Інші аксіоматизації числення висловлювань можна знайти в [28, 33].

5. ТЕОРЕМА ДЕДУКЦІЇ

Наведена нижче теорема, яка носить називу теореми дедукції, важлива при доведенні теорем, оскільки, по суті, вона дає нове правило виведення.

Теорема дедукції. Якщо Γ — множина формул, A і B — формули і $\Gamma, A \vdash B$, то $\Gamma \vdash A \rightarrow B$. Зокрема, якщо $A \vdash B$, то $\vdash A \rightarrow B$.

Доведення. Нехай B_1, B_2, \dots, B_n — виведення із $\Gamma \cup \{A\}$ формули B , тобто $B_n = B$. Доведення будемо вести індукцією по i — довжині виведення ($1 \leq i \leq n$) формули B .

При $i = 1$ B_1 повинна бути або:

- а) елементом Γ ;
- б) аксіомою;
- в) самою формuloю A .

У випадках “а” і “б” теорема випливає з того, що за схемою А1 маємо $B_1 \rightarrow (A \rightarrow B_1)$. Звідси за МР одержуємо, що $\Gamma \vdash (A \rightarrow B_1)$.

У випадку “в”, тобто коли $A = B_1$, із доведеного вище маємо, що $A \rightarrow A$ — теорема. Отже, $\Gamma \vdash A \rightarrow A$ і випадок $i = 1$ цим вичерпано.

Нехай тепер $\Gamma \vdash A \rightarrow B_k$ справедливе для всіх $k < i$. Покажемо, що $\Gamma \vdash A \rightarrow B_i$. У даному випадку можливі такі варіанти:

- а) B_i є аксіомою;
- б) B_i є елементом Γ ;
- в) $B_i = A$;
- г) B_i є наслідком за правилом МР деяких формул B_j і B_m ($j, m < i$), і B_m має вигляд $B_j \rightarrow B_i$.

Доведення випадків “а”—“в” нічим не відрізняється від доведення цих випадків для $i = 1$.

У випадку “г”, використовуючи принцип індукції, маємо $\Gamma \vdash \vdash A \rightarrow B_j$ і $\Gamma \vdash A \rightarrow (B_j \rightarrow B_i)$. За схемою аксіом А2 маємо $(A \rightarrow \rightarrow (B_j \rightarrow B_i)) \rightarrow ((A \rightarrow B_j) \rightarrow (A \rightarrow B_i))$. Отже, за правилом МР одержуємо $\Gamma \vdash (A \rightarrow B_j) \rightarrow (A \rightarrow B_i)$ і знову за правилом МР — $\Gamma \vdash A \rightarrow B_i$.

Теорема доведена.

Приклад 3.2.8

Довести теореми.

$$1. A \rightarrow B, B \rightarrow C \vdash A \rightarrow C.$$

Доведення.

- (1) $A \rightarrow B$ — гіпотеза;
- (2) $B \rightarrow C$ — гіпотеза;
- (3) A — гіпотеза;

- (4) B — за MP із (1), (3);
(5) C — за MP із (2), (4).

Отже, $A \rightarrow B$, $B \rightarrow C$, $A \vdash C$, і за теоремою дедукції маємо $A \rightarrow B$, $B \rightarrow C \vdash A \rightarrow C$.

Ця теорема показує, що операція імплікації є транзитивним відношенням.

2. $A \rightarrow (B \rightarrow C)$, $B \vdash A \rightarrow C$.

Д о в е д е н н я.

- (1) $A \rightarrow (B \rightarrow C)$ — гіпотеза;
(2) B — гіпотеза;
(3) A — гіпотеза;
(4) $B \rightarrow C$ — за MP із (1), (3);
(5) C — за MP із (4), (2).

Отже, $A \rightarrow (B \rightarrow C)$, B , $A \vdash C$ і за теоремою дедукції маємо

$$A \rightarrow (B \rightarrow C), B \vdash A \rightarrow C.$$

3. $\neg\neg B \rightarrow B$.

Д о в е д е н н я.

- (1) $(\neg B \rightarrow \neg\neg B) \rightarrow ((\neg B \rightarrow \neg B) \rightarrow B)$ — схема аксіом A3;
(2) $\neg B \rightarrow \neg B$ — теорема 1 із прикладу 3.2.7;
(3) $(\neg B \rightarrow \neg\neg B) \rightarrow B$ — теорема 2 із даного прикладу ;
(4) $\neg\neg B \rightarrow (\neg B \rightarrow \neg\neg B)$ — схема аксіом A1;
(5) $\neg\neg B \rightarrow B$ — теорема 1 із даного прикладу .

4. $B \rightarrow \neg\neg B$.

Д о в е д е н н я.

(1) $(\neg\neg\neg B \rightarrow \neg B) \rightarrow ((\neg\neg\neg B \rightarrow B) \rightarrow \neg\neg B)$ — схема аксіом A3;

- (2) $\neg\neg\neg B \rightarrow \neg B$ — теорема 3 із даного прикладу ;
(3) $(\neg\neg\neg B \rightarrow B) \rightarrow \neg\neg B$ — за MP із (1), (2);
(4) $B \rightarrow (\neg\neg\neg B \rightarrow B)$ — схема аксіом A1;
(5) $B \rightarrow \neg\neg B$ — теорема 1 із даного прикладу і за MP із (3),

(4).

Співставляючи теореми 3 і 4, одержуємо таку теорему:

$$B \Leftrightarrow \neg\neg B,$$

яка виражає відомий нам з булевих алгебр закон подвійного заперечення.

5. $\neg A \rightarrow (A \rightarrow B)$.

Д о в е д е н н я.

- (1) $\neg A$ — гіпотеза;
(2) A — гіпотеза;
(3) $A \rightarrow (\neg B \rightarrow A)$ — схема аксіом A1;

- (4) $\neg B \rightarrow A$ — за MP із (2), (3);
- (5) $\neg A \rightarrow (\neg B \rightarrow \neg A)$ — схема аксіом A1;
- (6) $\neg B \rightarrow \neg A$ — за MP із (5), (1);
- (7) $(\neg B \rightarrow \neg A) \rightarrow ((\neg B \rightarrow A) \rightarrow B)$ — схема аксіом A3;
- (8) $(\neg B \rightarrow A) \rightarrow B$ — за MP із (6), (7);
- (9) B — за MP із (4), (8).

Отже, $\neg A, A \vdash B$ і за теоремою дедукції маємо $\vdash \neg A \rightarrow (A \rightarrow B)$.

6. $(\neg B \rightarrow \neg A) \rightarrow (A \rightarrow B)$.

Д о в е д е н н я.

- (1) $\neg B \rightarrow \neg A$ — гіпотеза;
- (2) A — гіпотеза;
- (3) $(\neg B \rightarrow \neg A) \rightarrow ((\neg B \rightarrow A) \rightarrow B)$ — схема аксіом A3;
- (4) $A \rightarrow (\neg B \rightarrow A)$ — схема аксіом A1;
- (5) $(\neg B \rightarrow A) \rightarrow B$ — за MP із (1), (3);
- (6) $A \rightarrow B$ — теорема 1 із даного прикладу;
- (7) B — за MP із (2), (6).

Отже, $\neg B \rightarrow \neg A, A \vdash B$ і за теоремою дедукції маємо

$$\vdash (\neg B \rightarrow \neg A) \rightarrow (A \rightarrow B).$$

7. $(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$.

Д о в е д е н н я.

- (1) $A \rightarrow B$ — гіпотеза;
- (2) $\neg \neg A \rightarrow A$ — теорема 3 із даного прикладу;
- (3) $\neg \neg A \rightarrow B$ — теорема 1 із даного прикладу;
- (4) $B \rightarrow \neg \neg B$ — теорема 4 із даного прикладу;
- (5) $\neg \neg A \rightarrow \neg \neg B$ — теорема 1 із даного прикладу;
- (6) $(\neg \neg A \rightarrow \neg \neg B) \rightarrow (\neg B \rightarrow \neg A)$ — теорема цього прикладу;
- (7) $\neg B \rightarrow \neg A$ — за MP із (5), (6).

Отже, $A \rightarrow B \vdash \neg B \rightarrow \neg A$ і за теоремою дедукції маємо

$$\vdash (A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A).$$

Співставляючи теореми 6 і 7, одержуємо таку теорему:

$$A \rightarrow B \Leftrightarrow \neg B \rightarrow \neg A,$$

на якій ґрунтуються метод доведення тверджень, відомий під на-
звою *метод доведення від супротивного*.

8. $A \rightarrow (\neg B \rightarrow \neg(A \rightarrow B))$.

Д о в е д е н н я. Оскільки $A, A \rightarrow B \vdash B$, то за теоремою дедукції маємо $A \rightarrow ((A \rightarrow B) \rightarrow B)$. За теоремою 7 маємо

$$\vdash ((A \rightarrow B) \rightarrow B) \rightarrow (\neg B \rightarrow \neg(A \rightarrow B)).$$

І, нарешті, за теоремою 1 із даного прикладу одержуємо шукане

$$\vdash A \rightarrow (\neg B \rightarrow \neg(A \rightarrow B)).$$

9. $(A \rightarrow B) \rightarrow ((\neg A \rightarrow B) \rightarrow B)$.

Д о в е д е н н я.

- (1) $A \rightarrow B$ — гіпотеза;
- (2) $\neg A \rightarrow B$ — гіпотеза;
- (3) $(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$ — теорема 7 із даного прикладу;
- (4) $\neg B \rightarrow \neg A$ — за MP із (1), (3);
- (5) $(\neg A \rightarrow B) \rightarrow (\neg B \rightarrow \neg \neg A)$ — теорема 7 із даного прикладу;
- (6) $\neg B \rightarrow \neg \neg A$ — за MP із (2), (5);
- (7) $(\neg B \rightarrow \neg \neg A) \rightarrow (\neg B \rightarrow \neg A) \rightarrow B$ — схема аксіом А3;
- (8) $(\neg B \rightarrow \neg A) \rightarrow B$ — за MP із (6), (7);
- (9) B — за MP із (4), (8).

Отже, $A \rightarrow B, \neg A \rightarrow B \vdash B$ і за теоремою дедукції маємо

$$(A \rightarrow B) \rightarrow ((\neg A \rightarrow B) \rightarrow B). \blacktriangleleft$$

6. НЕСУПЕРЧНІСТЬ І ПОВНОТА ЧИСЛЕННЯ ВИСЛОВЛЮВАНЬ

Теорія L називається *несуперичною теорією*, якщо або A , або $\neg A$ є теоремами теорії Th , тобто існує виведення в Th або A , або $\neg A$.

Теорема 3.2.4. *Формула $A \in Th$ тоді і тільки тоді, коли A — тавтологія.*

Д о в е д е н н я. Для доведення теореми в один бік використовується той факт, що аксіоми А1–А3 є тавтологіями, в чому неважко переконатися. А застосування правила MP до тавтологій приводить знову до формули, яка є тавтологією. Отже, всяка теорема Th — тавтологія.

У другий бік теорема доводиться складніше і потребує доведення допоміжного твердження.

Лема 3.2.1. *Нехай A — формула, а B_1, B_2, \dots, B_k — пропозиційні змінні, що входять до формули A , і нехай задана деяка інтерпретація h . Покладемо B'_i рівним B_i , якщо $h(B_i) = 1$, і рівним $\neg B_i$, якщо $h(B_i) = 0$, і, нарешті, A' рівним A , коли $h(A) = 1$, і рівним $\neg A$, коли $h(A) = 0$.*

Тоді $B'_1, B'_2, \dots, B'_k \vdash A'$.

Д о в е д е н н я ведеться індукцією за числом n входжень у формулу A логічних зв'язок (звичайно припускається, що формула A записана без скорочень).

Якщо $n = 0$, то A — пропозиційна буква B , і твердження леми зводиться до $B \vdash B$ або $\neg B \vdash \neg B$. Отже, при $n = 0$ лема вірна. Припустимо, що лема справедлива при всіх $j < n$.

Випадок 1. A має вигляд заперечення — $\neg B$. Число входжень логічних зв'язок в B , очевидно, менше за n .

Випадок 1а. Нехай $h(B) = 1$, тоді $h(A) = 0$. Таким чином, $B' \in B$, а $A' \in \neg A$. За припущенням індукції, яке стосується формулі B , маємо $B'_1, B'_2, \dots, B'_k \vdash B$.

За теоремою 4 з прикладу 3.2.8 і MP отримуємо

$$B'_1, B'_2, \dots, B'_k \vdash \neg\neg B,$$

але $\neg\neg B$ якраз і є формулою A' .

Випадок 1б. Нехай $h(B) = 0$, тоді $B' \in \neg B$, а A збігається з A' . За припущенням індукції маємо

$$B'_1, B'_2, \dots, B'_k \vdash \neg B,$$

що і потрібно було довести, оскільки $\neg B$ і є A' .

Випадок 2. Нехай A має вигляд $B \rightarrow C$. Тоді число входжень логічних зв'язок в B і C менше, ніж в A . За припущенням індукції

$$\begin{aligned} B'_1, B'_2, \dots, B'_k &\vdash B, \\ B'_1, B'_2, \dots, B'_k &\vdash C. \end{aligned}$$

Випадок 2а. Нехай $h(B) = 0$, тоді $h(A) = 1$ і $B' \in \neg B$, а $A' \in A$. За припущенням індукції, маємо

$$B'_1, B'_2, \dots, B'_k \vdash \neg B$$

і з теореми 5 прикладу 3.2.8 одержуємо

$$B'_1, B'_2, \dots, B'_k \vdash B \rightarrow C,$$

але $B \rightarrow C$ і є формулою A .

Випадок 2б. Нехай $h(C) = 1$ і $h(A) = 1$, тоді $C' \in C$, а $A' \in A$. За припущенням індукції маємо

$$B'_1, B'_2, \dots, B'_k \vdash C$$

і із схеми аксіом A1 одержуємо

$$B'_1, B'_2, \dots, B'_k \vdash B \rightarrow C,$$

де $B \rightarrow C$ збігається з A .

Випадок 2в. Нехай $h(B) = 1$ і $h(C) = 0$, тоді $h(A) = 0$ і $A' \in \neg A$, $B' \in B$ і $C' \in \neg C$. За припущенням індукції маємо

$$\begin{aligned} B'_1, B'_2, \dots, B'_k &\vdash B, \\ B'_1, B'_2, \dots, B'_k &\vdash C, \end{aligned}$$

а звідси за теоремою 8 із прикладу 3.2.8 одержуємо

$$B'_1, B'_2, \dots, B'_k \vdash \neg(B \rightarrow C),$$

де $\neg(B \rightarrow C)$ і є A' .

Доведення леми закінчено.

Тепер завершимо доведення теореми 3.2.4.

Нехай A — тавтологія і B_1', B_2', \dots, B_k' — пропозиційні букви, які входять у формулу A . При будь-якій інтерпретації букв B_i внаслідок леми 3.2.1 маємо

$$B_1', B_2', \dots, B_k' \vdash A$$

(A' збігається з A , оскільки A — тавтологія). Завдяки цьому у випадку, коли B_k приймає значення 1, застосовуючи лему 3.2.1, одержуємо

$$B_1', B_2', \dots, B_k' \vdash A,$$

а коли B_k приймає значення 0, то за тією ж лемою одержуємо

$$B_1', B_2', \dots, \neg B_k' \vdash A.$$

Звідси за теоремою дедукції маємо

$$\begin{aligned} B_1', B_2', \dots, B_{k-1}' &\vdash B_k \rightarrow A, \\ B_1', B_2', \dots, B_{k-1}' &\vdash \neg B_k \rightarrow A. \end{aligned}$$

За теоремою 9 із прикладу 3.2.8 маємо

$$B_1', B_2', \dots, B_{k-1}' \vdash A.$$

Діючи таким чином і далі, виключаємо всі B_i із розгляду і приходимо після k кроків до $\vdash A$.

Теорема 3.2.4 доведена.

Тепер ця теорема дає можливість довести твердження, обернене до теореми дедукції.

Теорема 3.2.5. Якщо $\vdash A \rightarrow (B \rightarrow \dots \rightarrow (C \rightarrow D) \dots)$, то $A, B, \dots, C \vdash D$.

Д о в е д е н н я. Якщо $A \rightarrow (B \rightarrow \dots \rightarrow (C \rightarrow D) \dots)$ — теорема, то нехай C_1, C_2, \dots, C_k — її виведення. Тоді виведення

$$C_1, C_2, \dots, C_k, A \rightarrow (B \rightarrow \dots \rightarrow (C \rightarrow D) \dots),$$

A — гіпотеза,

$B \rightarrow (\dots \rightarrow (C \rightarrow D) \dots)$ — за MP із попередніх двох формул,

B — гіпотеза,

...

$C \rightarrow D$ — за MP із попередніх двох формул,

C — гіпотеза,

D є виведенням формули D із A, B, \dots, C , тобто $A, B, \dots, C \vdash D$.

Наслідок 3.2.1. Формула D є наслідком формул A, B, \dots, C тоді і тільки тоді, коли $A \& B \& \dots \& C \rightarrow D$ — тавтологія.

Дійсно, якщо $A \& B \& \dots \& C \rightarrow D$ — тавтологія, то $A \& B \& \dots \& C \rightarrow D = \neg(A \& B \& \dots \& C) \vee D = \neg A \vee \neg B \vee \dots \vee \neg C \vee D = \neg A \vee \neg B \vee \dots \vee (C \rightarrow D) = \dots = A \rightarrow (B \rightarrow \dots \rightarrow (C \rightarrow D) \dots)$.

Остання формула — тавтологія. За теоремою 3.2.5 маємо $A, B, \dots, C \models D$.

Наслідок 3.2.2. Формула D є наслідком формул A, B, \dots, C тоді і тільки тоді, коли $A \& B \& \dots \& C \& \neg D$ — суперечність.

Доведення пропонується як проста вправа.

Приклад 3.2.9

Довести, що $\neg P$ є логічним наслідком формул $P \rightarrow Q$ і $\neg Q$.

Розв'язок. За наслідком 3.2.1 формула $(P \rightarrow Q) \& \neg Q \rightarrow \neg P$ має бути тавтологією. Користуючись таблицями істинності, знаходимо:

P	Q	$\neg P$	$\neg Q$	$P \rightarrow Q$	$(P \rightarrow Q) \& \neg Q$	$(P \rightarrow Q) \& \neg Q \rightarrow \neg P$
1	1	0	0	1	0	1
1	0	0	1	0	0	1
0	1	1	0	1	0	1
0	0	1	1	1	1	1

Отже, $\neg P$ є наслідком формул $P \rightarrow Q$ і $\neg Q$. \blacksquare

Теорема 3.2.4 віправдовує аксіоматизацію числення висловлювань, оскільки вона стверджує, що синтаксичний наслідок (наслідок, виведений з аксіом) є також і семантичним наслідком (тобто тавтологією), і навпаки.

Теорема 3.2.6. Теорія \mathcal{Th} числення висловлювань є несуперечною теорією.

Доведення теореми є простим наслідком теореми 3.2.4. Дійсно, якщо $A (\neg A)$ — теорема числення висловлювань, то $A (\neg A)$ повинна бути тавтологією. Отже, $\neg A (A)$ не може бути тавтологією і за теоремою 3.2.4 не є теоремою числення висловлювань.

7. ЧИСЛЕННЯ ВИСЛОВЛЮВАНЬ І БУЛЕВІ АЛГЕБРИ

Розглядаючи числення висловлювань \mathcal{Th} в цілому, неважко помітити аналогії між булевою алгеброю та численням висловлювань (між операціями, між функціями істинності і виразами булевої алгебри і т.д.). З'ясуємо, наскільки глибокі ці аналогії.

Насамперед відзначимо, що множину S всіх формул числення висловлювань за теоремою 3.2.2 можна розглядати як універсальну алгебру з двома операціями — унарною \neg і бінарною $\&$ та носієм S .

Крім того, за допомогою зв'язки \Leftrightarrow можна ввести відношення еквівалентності R на множині формул S : $A R B$ в тому і тільки в

тому випадку, коли $A \Leftrightarrow B$. Інакше кажучи, $A R B \Leftrightarrow h(A) = h(B)$ для будь-якої інтерпретації h . Те, що відношення R є відношенням еквівалентності, очевидно, але неважко переконатися, що відношення R є конгруентністю. Дійсно, якщо A, B, C — формули числення висловлювань, то із $A R B$ випливає справедливість формул $(\neg A) R (\neg B)$ і $(A \& C) R (B \& D)$. Дійсно, якщо $A R B$ і $C R D$, то $h(A) = h(B)$, $h(C) = h(D)$ і

$$h(\neg A) = \neg h(A) = \neg h(B) = h(\neg B), \\ h(A \& C) = h(A) \& h(C) = h(B) \& h(D) = h(B \& D).$$

Отже, відношення R є конгруентністю на множині формул S , і можна розглядати фактор-алгебру Th/R . Алгебра Th/R називається **алгеброю Лінденаума числення висловлювань**, або **алгеброю висловлювань**.

Основну властивість алгебри Лінденаума дає така теорема.

Теорема 3.2.7. *Алгебра $Th/R = ([S], \{\neg, \&\})$, де $[S]$ — множина класів еквівалентних формул, є булевою алгеброю.*

Д о в е д е н н я. Поставимо у відповідність 0 булевої алгебри формулу $A \& \neg A$, а 1 — формулу $\neg(A \& \neg A)$. Зауважимо, що булеву алгебру, за законами де Моргана, можна розглядати теж як алгебру з двома операціями: $\&$ і \neg . Після цього доведення теореми зводиться до простої перевірки виконання законів булевої алгебри, тобто до перевірки, що $(A \& B) R (B \& A)$ і т.д. Виконання цієї перевірки пропонується як вправа.

З теореми 3.2.7 випливає, що числення висловлювань являє собою модель булевої алгебри.

§ 3.3. МЕТОДИ ПЕРЕВІРКИ ТОТОЖНОЇ ІСТИННОСТІ ФОРМУЛ ЧИСЛЕННЯ ВИСЛОВЛЮВАНЬ

З попереднього параграфа випливає простий метод перевірки заданої формулі на тотожність, тобто чи є вона тавтологією, чи суперечністю. Для цього потрібно знайти її значення при всіх можливих інтерпретаціях. Будемо називати цей метод **тривіальним**. Хоча він досить простий, але основний його недолік — громіздкість. Існують і більш досконалі методи. Розглянемо деякі з них.

1. АЛГЕБРАЇЧНИЙ МЕТОД

Алгебраїчний метод ґрунтуються на застосуванні законів булевої алгебри (згідно з теоремою 3.2.7) для спрощення формул логіки числення висловлювань. Оскільки різні формули можуть набувати одних і тих самих значень, тобто можуть бути логічно еквівалентними, то краще вибрати для відповідної перевірки ту з формул, яка простіша. Прикладами такого роду перетворень можуть служити алгоритми побудови ДНФ і КНФ. Крім того, якщо P — деяка скінченна множина формул, яка складається із n формул, то, користуючись формулами з P , можна побудувати нескінчу-
нну множину формул, серед яких існує лише 2^n логічно різних формул. Алгебраїчний метод дає можливість виділяти цю множину формул, тобто замінити розгляд нескінченної множини формул скінченною множиною.

2. МЕТОД КУАЙНА

Метод Куайна являє собою безпосереднє узагальнення тривіального алгоритму.

Нехай $\{p, q, \dots, r\}$ — упорядкована множина висловлювань, які зустрічаються у формулі $P(p, q, \dots, r)$. Візьмемо перше з висловлювань — p і припишемо йому, наприклад, значення 1 (0). Підставимо це значення у формулу P і виконаємо обчислення, які можуть виявитися необхідними внаслідок такої підстановки. Після виконання обчислень одержуємо деяку формулу $P'(q, \dots, r)$, до якої застосовується описана процедура, тобто вибираємо формулу q , приписуємо їй значення 1 (0), виконуємо обчислення і т.д. Може трапитися так, що на деякому кроці буде одержана формула P'' , яка є тавтологією або суперечністю незалежно від значень висловлювань, які входять до складу формули P'' . Отже, на цьому кроці роботу алгоритму можна зупинити. Таким чином, метод Куайна в деяких випадках приводить до розгляду значно меншої кількості інтерпретацій, ніж тривіальний алгоритм.

Приклад 3.3.1

Довести, що формула

$$P = (((p \& q) \rightarrow r) \& (p \rightarrow q)) \rightarrow (p \rightarrow r)$$

є теоремою числення висловлювань.

Розв'язок. Множина висловлювань формули $P \in \{p, q,$

$r\}$. Вибираємо висловлювання p . При цьому може бути два випадки.

1. $p = 1$. Тоді

$$P = (((1 \& q) \rightarrow r) \& (1 \rightarrow q)) \rightarrow (1 \rightarrow r) = ((q \rightarrow r) \& q) \rightarrow r = P'.$$

Вибираємо тепер q і розглядаємо знову можливі випадки:

a1) $q = 1$, тоді $P' = ((1 \& r) \& 1) \rightarrow r = r \rightarrow r = r$ — тавтологія;

a2) $q = 0$, тоді $P' = ((0 \rightarrow r) \& 0) \rightarrow r = 0 \rightarrow r = 1$.

2. $p = 0$. Тоді

$$\begin{aligned} P &= (((0 \& q) \rightarrow r) \& (0 \rightarrow q)) \rightarrow (0 \rightarrow r) = \\ &= ((0 \rightarrow r) \& 1) \rightarrow 1 = 1 \rightarrow 1 = 1. \end{aligned}$$

Отже, дана формула є тавтологією, а значить, і теоремою числення висловлювань. Крім того, можливі інтерпретації формулі r в даній формулі P не відіграють ніякої ролі. \blacktriangleleft

3. МЕТОД РЕДУКЦІЇ

Метод редукції дає можливість виконувати перевірку формул числення висловлювань шляхом зведення до абсурду. Він особливо зручний, коли в записі формулі зустрічається багато імплікацій.

Нехай формула P має вигляд імплікації, наприклад $P = Q \rightarrow \rightarrow R$. Припустимо, що в деякій інтерпретації h формула P приймає значення 0. Тоді відповідно з таблицею істинності для імплікації маємо $h(Q) = 1$ і $h(R) = 0$. Таким чином, перевірка формули P зводиться до перевірки формул Q і R . Після цього даний процес застосовується до формул Q і R і т.д.

Приклад 3.3.2

Чи є формула

$$P = ((p \& q) \rightarrow r) \& (p \rightarrow (q \rightarrow r))$$

тавтологією?

Розв'язок. Нехай для деякої інтерпретації h маємо $h(P) = 0$. Тоді

$$h(p \rightarrow (q \rightarrow r)) = 0 \text{ і } h((p \& q) \rightarrow r) = 1.$$

Застосуємо цю ж процедуру до першої з формул. Одержано

$$h(p) = 1 \text{ і } h(q \rightarrow r) = 0.$$

Звідси знаходимо, що

$$h(p) = 1, h(q) = 1 \text{ і } h(r) = 0.$$

Але отримані значення суперечать тому, що

$$h((p \ \& \ q) \rightarrow r) = 1.$$

Одержана суперечність показує, що формула P — тавтологія. \blacktriangleleft

4. МЕТОД ДЕВІСА—ПАТНЕМА

Перш ніж перейти до викладення метода Девіса—Патнема, введемо деякі означення.

Літерою називається атом або заперечення атома. **Диз'юнктом** називається диз'юнкція літер. **Однічним диз'юнктом** називається однолітерний диз'юнкт. Якщо диз'юнкт не має жодної літери, то він називається **пустим диз'юнктом** і позначається 0. Літери L і $\neg L$ називаються **контрарніми**.

Нехай S — множина диз'юнктів. Суть методу Девіса—Патнема полягає в застосуванні таких чотирьох правил.

ДП1. Викреслити всі тавтологічні диз'юнкти із S . Множина диз'юнктів S' , що залишилася, суперечна тоді і тільки тоді, коли S суперечна. (**Правило тавтології**.)

ДП2. Якщо існує одиничний диз'юнкт L в S , то S' одержано з S шляхом викреслення з S тих диз'юнктів, які містять L . Якщо S' пустий, то S — істина. В протилежному випадку будуємо множину S'' шляхом вилучення з S' всіх входжень $\neg L$. S'' суперечна тоді і тільки тоді, коли і S суперечна. Зауважимо, що, коли $\neg L$ — одиничний диз'юнкт, то при викресленні $\neg L$ він перетворюється в пустий диз'юнкт. (**Правило однолітерних диз'юнктів**.)

ДП3. Літера L деякого диз'юнкта з S називається чистою в S тоді і тільки тоді, коли літера $\neg L$ не з'являється ні в якому диз'юнкті з S . Якщо літера L чиста в S , то викреслимо всі диз'юнкти, які містять L . Множина S' , що залишилася, суперечна тоді і тільки тоді, коли і S суперечна. (**Правило чистих літер**.)

ДП4. Якщо множину S можна подати у вигляді

$$(A_1 \vee L) \ \& \ \dots \ \& \ (A_m \vee L) \ \& \ (B_1 \vee \neg L) \ \& \ \dots \ \& \ (B_n \vee \neg L) \ \& \ R,$$

де A_i , B_i і R чисті від L і $\neg L$, $S_1 = A_1 \ \& \ \dots \ \& \ A_m \ \& \ R$ і $S_2 = B_1 \ \& \ \dots \ \& \ B_n \ \& \ R$, то S суперечна тоді і тільки тоді, коли $(S_1 \vee S_2)$ суперечна. (**Правило розщеплення**.)

Теорема 3.3.1. *Метод Девіса—Патнема є коректним.*

Доведення.

Правило 1. Оскільки тавтологія виконується при будь-якій інтерпретації, то S' — суперечність тоді і тільки тоді, коли і S — суперечність.

Правило 2. Якщо S' — пуста множина диз'юнктів, то всі диз'юнкти з S включають L . Значить, всяка інтерпретація, яка включає L , може задовольняти S . Отже, S виконується в цій інтерпретації. Покажемо тепер, що S'' суперечна тоді і тільки тоді, коли S суперечна. Припустимо, що S'' суперечна. Якщо S несуперечна, то існує модель M для S , яка включає L . Для S'' модель M повинна задовольняти всі диз'юнкти, які не включають L . Тепер оскільки на M літера $\neg L$ суперечна, то моделі M повинні задовольняти всі диз'юнкти, які на початку включали $\neg L$. Отже, S'' виконується на моделі M . А це суперечить тому, що S'' не має моделі. Таким чином, S теж повинна бути суперечністю.

Навпаки, нехай S — суперечність. Якщо S'' виконується при деякій інтерпретації, то існує модель M'' для S'' . Звідси випливає, що всяка інтерпретація, яка включає модель M'' і L , повинна служити моделлю для S' . А це суперечить тому, що S не має моделі. Отже, S'' повинна бути суперечністю. Значить, S суперечна тоді і тільки тоді, коли S'' суперечна.

Правило 3. Припустимо, що S' — суперечність. Тоді і S буде суперечністю, оскільки S' — підмножина множини S . Навпаки, нехай S — суперечність. Якщо S' виконується при деякій інтерпретації, то S' має модель M . Значить, ні L , ні $\neg L$ не знаходяться в S' , і ні L , ні $\neg L$ не знаходяться в M . Звідси випливає, що всяка інтерпретація S , яка включає M і L , є моделлю для S . А це суперечить тому, що S не має моделі. Отже, S повинна бути суперечністю. Таким чином, S' — суперечність тоді і тільки тоді, коли S — суперечність.

Правило 4. Припустимо, що S — суперечність. Якщо $S_1 \vee S_2$ виконується при деякій інтерпретації, то або S_1 , або S_2 має модель. Коли S_1 (S_2) має модель M , то всяка інтерпретація S , яка включає $\neg L$ (L), є моделлю для S . А це суперечить припущення. Значить, $S_1 \vee S_2$ — суперечність. Припустимо тепер, що $S_1 \vee S_2$ — суперечність. Якщо S виконується при деякій інтерпретації, то S має модель M . Коли M включає $\neg L$ (L), то M може задовольняти S_1 (S_2), а це суперечить тому, що $S_1 \vee S_2$ не має моделі. Отже, S повинна бути суперечністю. Таким чином, S суперечна тоді і тільки тоді, коли $S_1 \vee S_2$ суперечна.

Теорема доведена.

Приклади 3.3.3

- Довести, що $S = (P \vee Q \vee \neg R) \& (P \vee \neg Q) \& \neg P \& R \& U$ суперечна.

Розв'язок.

- (1) $(P \vee Q \vee \neg R) \& (P \vee \neg Q) \& \neg P \& R \& U$,
- (2) $(Q \vee \neg R) \& (\neg Q) \& R \& U$ — правило ДП2 з $\neg P$,
- (3) $\neg R \& R \& U$ — правило ДП2 з $\neg Q$,
- (4) $0 \& U$ — правило ДП2 з $\neg R$.

Оскільки останній диз'юнкт включає пустий диз'юнкт 0, то формула S суперечна.

2. Показати, що $S = (P \vee Q) \& \neg Q \& (\neg P \vee Q \vee R)$ несуперечна.

Розв'язок.

- (1) $(P \vee Q) \& \neg Q \& (\neg P \vee Q \vee R)$,
- (2) $P \& (\neg P \vee \neg R)$ — правило ДП2 з $\neg Q$,
- (3) $\neg R$ — правило ДП2 з P ,
- (4) 0 — правило ДП2 з $\neg R$.

Остання множина являє собою пустий диз'юнкт. Отже, S несуперечна. ▲

5. МЕТОД РЕЗОЛЮЦІЙ

Метод резолюцій — це, по суті, узагальнення правила однолітерних диз'юнктів Девіса—Патнема.

Розглянемо диз'юнкти виду

$C : P$,

$C' : \neg P \vee Q$.

Користуючись правилом однолітерних диз'юнктів, із C і C' можна одержати диз'юнкт $C'' : Q$.

Узагальнюючи дане правило і застосовуючи його до будь-якої пари диз'юнктів (не обов'язково лише одиничних), одержуємо правило резолюції (ПР).

Правило резолюції. Для будь-яких диз'юнктів C і C' , якщо існує літера L в C , контрапарна літера L' в C' , то, викресливши L і L' із C і C' відповідно, побудуємо диз'юнкцію членів, що залишилися. Побудований диз'юнкт називається резольвентою C і C' .

Важливу властивість резольвенти двох диз'юнктів — C_1 і C_2 — виявляє таке твердження.

Теорема 3.3.2. Нехай дано два диз'юнкти: C_1 і C_2 . Тоді резольвента C диз'юнктів C_1 і C_2 — логічний наслідок C_1 і C_2 .

Доведення. Нехай $C_1 = L \vee C'_1$, $C_2 = \neg L \vee C'_2$, $C = C'_1 \vee C'_2$, де C'_1 , C'_2 — диз'юнкції літер. Припустимо, що C_1 і C_2 виконуються при інтерпретації h . Покажемо, що C також виконується при інтерпретації h . Нехай L хибна в h . Тоді C_1 не може бути одиничним диз'юнктом, оскільки C_1 була б хибною в h . Отже, C'_1 повинна виконуватись в h , а тоді і $C = C'_1 \vee C'_2$ вико-

нується в h . Таким чином, $C_1' \vee C_2'$ повинна виконуватись при інтерпретації h .

Теорема доведена.

Означення 3.3.1. Нехай S — множина диз'юнктів. Резолютивним виведенням диз'юнкта C із S називається така скінченна послідовність C_1, C_2, \dots, C_k диз'юнктів, в якій кожний з C_i або належить S , або є резольвентою диз'юнктів, що передують C_i , і $C = C_k$.

Виведення пустого диз'юнкта 0 із S називається **доведенням суперечності** S , або **спростуванням** S .

Говорять, що диз'юнкт C може бути виведений або одержаний із S , якщо існує резолютивне виведення C із S .

Правило резолюції (ПР) — досить потужне правило виведення. В наступних параграфах даного розділу воно буде визначене для логіки предикатів першого порядку, де і покажемо повноту методу резолюцій, тобто, що дана множина диз'юнктів S суперечна тоді і тільки тоді, коли існує резолютивне виведення пустого диз'юнкта 0 із S . А зараз розглянемо деякі приклади застосування цього методу в логіці висловлювань.

Приклади 3.3.4

1. Перевірити суперечність множини диз'юнктів

$$S = \{P, \neg P \vee Q \vee R, \neg Q \vee C, \neg R \vee C, \neg C\}.$$

Розв'язок. Застосовуючи правило резолюції до S , маємо:

- | | |
|----------------------------|--------------------|
| (1) P | диз'юнкт 1, |
| (2) $\neg P \vee Q \vee R$ | диз'юнкт 2, |
| (3) $\neg Q \vee C$ | диз'юнкт 3, |
| (4) $\neg R \vee C$ | диз'юнкт 4, |
| (5) $\neg C$ | диз'юнкт 5, |
| (6) $Q \vee R$ | за ПР із (1), (2), |
| (7) $\neg R$ | за ПР із (4), (5), |
| (8) $\neg Q$ | за ПР із (3), (5), |
| (9) Q | за ПР із (6), (7), |
| (10) 0 | за ПР із (8), (9). |

Отже, задана множина диз'юнктів суперечна, оскільки з неї виводиться пустий диз'юнкт.

2. Перевірити суперечність множини диз'юнктів

$$S = \{P \vee Q, P \vee R, \neg Q \vee \neg R, \neg P\}.$$

Розв'язок. Застосовуючи ПР до S , маємо:

(1) $P \vee Q$	диз'юнкт 1,
(2) $P \vee R$	диз'юнкт 2,
(3) $\neg Q \vee \neg R$	диз'юнкт 3,
(4) $\neg P$	диз'юнкт 4,
(5) $P \vee \neg R$	за ПР із (1), (3),
(6) Q	за ПР із (1), (4),
(7) $P \vee \neg Q$	за ПР із (2), (3),
(8) R	за ПР із (2), (4),
(9) P	за ПР із (2), (5),
(10) $\neg R$	за ПР із (3), (6),
(11) $\neg Q$	за ПР із (3), (8),
(12) $\neg R$	за ПР із (4), (5),
(13) $\neg Q$	за ПР із (4), (7),
(14) 0	за ПР із (4), (9).

Слід зазначити, що перевірка множини диз'юнктів на суперечність за допомогою правила резолюції загалом недетермінована, оскільки будувати резольвенти можна різні. З наведеного прикладу видно, що описана стратегія застосування правила резолюції не є оптимальною. Виведення пустого диз'юнкта з множини S можна було б виконати з меншим числом резольвент, наприклад:

(5) Q	за ПР із (1), (4),
(6) R	за ПР із (2), (4),
(7) $\neg Q$	за ПР із (3), (6),
(8) 0	за ПР із (5), (7). ▲

Контрольні питання

- Що називається атомом?
- Дайте визначення ппф.
- Коли логічна теорія вважається визначеною?
- Яка формула називається:
 - тавтологією;
 - суперечністю?
- Скільки аксіом і правил виведення має числення висловлювань?
- Сформулюйте аксіоми числення висловлювань. Які ви знаєте інші аксіоматизації числення висловлювань?
- Скільки правил виведення ви знаєте в численні висловлювань?
- Який взаємозв'язок існує між булевими алгебрами і численням висловлювань?
- Яка теорія називається несуперечною?
- Сформулюйте теорему:
 - дедукції;
 - обернену до теореми дедукції.
- Що називається літерою?

12. Які літери називаються контрарними?
13. Що називається диз'юнктом?
14. В чому полягає суть методу резолюцій?
15. В чому полягає суть методу Куайна?
16. Назвіть правила виведення числення висловлювань, відмінні від правила МР.
17. Чи буде теоремою числення висловлювань формула $A \vee \neg A$?

Задачі і вправи

1. Користуючись таблицями істинності, спробуйте впевнитись у тому, що логічні зв'язки \neg , $\&$, \vee , які розглядаються як операції над формулами, задовольняють закони булевої алгебри.
2. Виясніть, чи є формулою числення висловлювань вираз:
 - a) $(A \& B) C \neg D$;
 - б) $(A \& B) \rightarrow C$;
 - в) $((A \rightarrow B) \& \neg B$;
 - г) $((\neg A) \rightarrow B) \rightarrow \neg(C \vee D)$.
3. Скількома способами можна розставити дужки в таких виразах:
 - а) $A \rightarrow B \vee \neg B \& C$;
 - б) $A \rightarrow B \rightarrow C \rightarrow \neg A \rightarrow \neg B$;
 - в) $A \& B \vee C \& D \& C \vee A ?$
4. Напишіть всі підформули формул:
 - а) $((A \rightarrow B) \& (B \rightarrow C)) \rightarrow (\neg A \vee C)$;
 - б) $((A \rightarrow B) \rightarrow ((A \rightarrow \neg B) \rightarrow \neg B))$.
5. Побудуйте таблиці істинності для таких формул:
 - а) $((P \rightarrow Q) \vee (P \rightarrow (Q \& P)))$;
 - б) $(\neg P \rightarrow \neg(Q \& P)) \rightarrow (P \vee R)$;
 - в) $((P \& (Q \rightarrow P)) \rightarrow \neg P)$;
 - г) $((P \& \neg Q) \rightarrow Q) \rightarrow (P \rightarrow Q)$;
 - д) $((P \rightarrow (Q \vee R)) \rightarrow ((P \rightarrow Q) \rightarrow (P \rightarrow R)))$;
 - е) $((P \& (Q \vee \neg P)) \& ((\neg Q \rightarrow P) \vee Q))$.
6. Доведіть, що існують інтерпретації, в яких виконуються формулі:
 - а) $\neg(P \rightarrow \neg P)$;
 - б) $((P \rightarrow Q) \rightarrow (Q \rightarrow P))$;
 - в) $((Q \rightarrow (P \& R)) \& \neg((P \vee R) \rightarrow Q))$.
7. Доведіть, що наведені нижче формулі є тавтологіями:
 - а) $((P \rightarrow Q) \vee (Q \rightarrow P))$;
 - б) $((P \rightarrow Q) \vee (P \rightarrow \neg Q))$;
 - в) $(P \rightarrow (Q \rightarrow (P \& Q)))$;
 - г) $((P \rightarrow Q) \rightarrow ((Q \rightarrow R) \rightarrow (P \rightarrow R)))$;
 - д) $((\neg P \rightarrow \neg Q) \rightarrow (Q \rightarrow P))$;
 - е) $(P \rightarrow (Q \rightarrow P))$;
 - ж) $P \vee \neg P$;
 - з) $((P \rightarrow Q) \rightarrow ((P \rightarrow (Q \rightarrow R)) \rightarrow (P \rightarrow R)))$;
 - і) $(P \rightarrow Q) \rightarrow P$;

- к) $P \rightarrow (P \vee Q)$;
 л) $Q \rightarrow (P \vee Q)$;
 м) $(P \vee P) \rightarrow P$;
 н) $(\neg P \rightarrow (P \rightarrow Q))$.

8. При яких значеннях змінних x, y, z, u, v, w наведені нижче формулі хибні:

- а) $((x \rightarrow (y \& z)) \rightarrow (\neg y \rightarrow \neg x)) \rightarrow \neg y$;
 б) $((x \& y) \vee (x \& z) \vee (y \& z) \vee (\neg x \& \neg z))$;
 в) $((x \vee y) \vee z) \rightarrow ((x \vee) \& (x \vee z)))$;
 г) $((x \vee y) \& ((y \vee z) \& (z \vee x))) \rightarrow ((x \& y) \& z))$;
 д) $((x \vee y) \rightarrow ((\neg x \& y) \vee (x \& \neg y)))$?

9. Доведіть, що:

- а) $\neg A \rightarrow B, A \rightarrow C \vdash \neg B \rightarrow C$ (ПР);
 б) $\neg A \rightarrow B, A \rightarrow C, B \rightarrow D \nvdash \neg C \rightarrow D$;
 в) $A \rightarrow B, C \rightarrow \neg B \vdash A \rightarrow \neg C$.

10. Доведіть, що формули ГА1)—ГА4) і РА1)—РА3) є теоремами в численні висловлювань, тобто що вони виводяться із аксіом А1)—А3) за допомогою правила виведення.

§ 3.4. ЧИСЛЕННЯ ПРЕДИКАТИВ ПЕРШОГО ПОРЯДКУ

Існують такі види логічних формул, які не можна записати у вигляді формул числення висловлювань. Наприклад:

- 1) всі риби, окрім акул, добре відносяться до дітей;
- 2) всі люди смертні. Сократ — людина. Значить, Сократ смертний;
- 3) кожен, хто дружить з Ярославом, є другом Ігоря. Олег не є другом Ігоря, значить, Олег не є другом Ярослава.

Коректність таких висновків ґрунтується не тільки на істинності відповідних функціональних відношень, а також і на розумінні таких слів, як “всі”, “всякий” і т. д.

Для того щоб зробити більш зрозумілою структуру складних висловлювань, користуються спеціальною мовою — мовою числення **предикатів першого порядку**. Ця мова є розширенням мови термів, до визначення якої ми і перейдемо.

1. АЛФАВІТ І ФОРМУЛИ

Розширимо алфавіт функціональних символів мови термів символами

$$A_1^{n_1}, A_2^{n_2}, \dots, A_k^{n_k}, \dots,$$

які будуть називатися *предикатними символами*. Верхній індекс предикатного символу, як і раніше, буде означати його арність. Множину предикатних символів будемо називати *сигнатурою предикатів* і позначати Π .

Предикатні символи, коли їх застосувати до термів, породжують елементарні (атомарні) формулі, або точніше: якщо A^n — предикатний символ, t_1, t_2, \dots, t_n — терми, то $A^n(t_1, t_2, \dots, t_n)$ — елементарна формула.

Означення формул числення предикатів першого порядку. Індуктивно це подається так:

- всяка елементарна формула є формулою;
- якщо A і B — формули і x — предметна змінна, то кожний із виразів $(\neg A)$, $(A \rightarrow B)$, $(\forall x)(A)$ є формулою;
- вираз є формулою тоді і тільки тоді, коли це випливає з пунктів а), б).

Означення формул $(A \& B)$, $(A \vee B)$, $(A \Leftrightarrow B)$ таке в численні висловлювань.

Символи \neg , \rightarrow , $\&$, \vee , \Leftrightarrow називаються *логічними зв'язками*, а \forall — *квантором загальності*. Часто ще розглядають квантор існування $(\exists x)(A)$, який визначається як $\neg(\forall x)(\neg A)$. Формула $(\forall x)(A)$ читається “для всіх x має місце A ”, а формула $(\exists x)(A)$ — “існує x , такий, що має місце A ”.

Зауважимо, що мають місце ранги логічних зв'язок, про які йшла мова в § 3.2. Умовимося, що квантори \forall і \exists розміщуються за рангами між зв'язками \Leftrightarrow , \rightarrow і \vee , $\&$, \neg . Умовимося також опускати дужки, в яких знаходитьсь формула $Q A$ у формулах виду $Q_1(Q A)$, де Q і Q_1 — довільні квантори. Наприклад, замість $(\forall x_1 (\exists x_2 (\forall x_4 A_1^3 (x_1, x_2, x_4))))$ пишеться формула $\forall x_1 \exists x_2 \forall x_4 A_1^3 (x_1, x_2, x_4)$.

Введемо поняття вільного і зв'язаного входження змінної в формулу. У виразі $\forall x A$ ($\exists x A$) формула A називається *областю дії квантора* $\forall x$ ($\exists x$). Слід зазначити, що формула A може і не мати входжень змінної x . У такому випадку вважається, що формулі A і $\forall x A$ однакові. Входження змінної x в дану формулу A називається *зв'язаним*, якщо x — змінна квантора $\forall x$ або $\exists x$, який входить у дану формулу або перебуває в області дії квантора $\forall x$ чи $\exists x$, який входить у дану формулу. В протилежному випадку входження змінної x в дану формулу називається *вільним*.

Змінна називається *вільною* (зв'язаною) змінною в даній формулі, якщо існують вільні (зв'язані) її входження в цю формулу. Отже, змінна може бути одночасно вільною і зв'язаною в одній і тій же формулі. Формула називається *замкнutoю*, якщо вона не має вільних змінних.

Приклад 3.4.1

Розглянемо формули:

- 1) $A_1^2(x, y)$;
- 2) $A_1^2(x, y) \rightarrow \forall x A_1^1(x)$;
- 3) $\forall x (A_1^2(x, y) \rightarrow \forall x A_1^1(x))$.

Єдине входження змінної x в формулу 1) вільне. Перше входження змінної x в формулі 2) вільне, а друге і третє зв'язані. Всі входження змінної x в формулу 3) зв'язані. Слід зауважити, що одна і та ж змінна може мати вільні і зв'язані входження в одну і ту ж формулу, як це було у формулі 2). Зазначимо також, що входження змінної може бути зв'язаним у тій чи іншій формулі A і водночас вільним у деякій підформулі формулі A . Наприклад, перше входження x у формулу 2) вільне, але формула 2) є підформулою формулі 3), де те саме входження x уже зв'язане. ▲

Терм t називається **вільним** для змінної x у формулі A , якщо жодне вільне входження x в A не лежить в області дії жодного квантора $\forall y$, де y — змінна, що входить в терм t .

Приклад 3.4.2

1. Терм x вільний для змінної y в $A^1(y)$, але не є вільним в $\forall x A^1(y)$. Терм $f(x, z)$ вільний для змінної x в $\forall y A^2(x, y) \rightarrow B^1(x)$, але не є вільним для x в $\exists z \forall y (A^2(x, y) \rightarrow B^1(x))$.

2. Всякий терм, який не має змінних (константний терм), вільний для всякої змінної в будь-якій формулі.

3. Терм t вільний для будь-якої змінної у формулі A , якщо жодна змінна терма t не є зв'язаною змінною у формулі A .

4. Терм x вільний для x в будь-якій формулі.

5. Всякий терм вільний для x у формулі A , коли A не містить у собі вільних входжень x . ▲

2. ІСТИННІСТЬ, ІНТЕРПРЕТАЦІЇ, МОДЕЛІ

Формули мають сенс тільки тоді, коли існує яка-небудь інтерпретація символів, що входять до цих формул. Переїдемо до визначення виконання і істинності формул числення, які будемо формулювати відповідно до індуктивних кроків означення формул.

Нехай маємо деяку інтерпретацію, тобто систему, яка складається з непустої множини D — області інтерпретації — і відповідності h , що ставить кожному предикатному символу A_j^n деяке n -арне відношення на D , кожному функціональному символу f_j^n — деяку n -арну операцію на D і кожній предметній константі a —

деякий елемент із D . При заданій інтерпретації предметні змінні пробігають область D цієї інтерпретації.

Приклад 3.4.3

Розглянемо формулі:

- 1) $A_1^2(x, y)$;
- 2) $\forall y A_1^2(x, y)$;
- 3) $\exists y \forall x A_1^2(y, x)$.

Якщо множина натуральних чисел \mathbb{N}^+ взята за область інтерпретації і $A_1^2(x, y)$ інтерпретується як $x \leq y$, то формула 1) являє собою відношення $x \leq y$, яке виконується на всіх упорядкованих парах чисел (a, b) , таких, що $a \leq b$; формула 2) являє собою відношення з одним аргументом x виду "для кожного натурального додатного числа y , $x \leq y'$ ", яке виконується лише для числа 1; формула 3) буде істинним висловлюванням, яке стверджує існування мінімального цілого додатного числа. Якби областью інтерпретації служила множина всіх цілих чисел, то формула 3) була б хибною. \blacktriangleleft

Нехай задана деяка область інтерпретації D , і нехай Σ — множина всіх зліченних послідовностей елементів із D . Визначимо тепер, що означає вислів «формула A виконана на послідовності $s = (b_1, b_2, \dots)$ із Σ при заданій інтерпретації».

Нехай s^* означає функцію одного аргументу, яка визначена на множині термів і приймає свої значення в області D деякої інтерпретації h , тобто $s^*: T(\Omega, X) \rightarrow D$. Ця функція визначається так:

- а) якщо терм t — предметна змінна x_i , то $s^*(t) = b_i$;
- б) якщо терм t — предметна константа, то $s^*(t) = h(t)$, тобто $s^*(t)$ збігається з інтерпретацією цієї константи в області D ;

в) якщо f^n — функціональний символ, якому при інтерпретації h відповідає операція g на області D , і t_1, t_2, \dots, t_n — терми, то $s^*(f^n(t_1, t_2, \dots, t_n)) = g(s^*(t_1), s^*(t_2), \dots, s^*(t_n))$.

Отже, якщо говорити неформально, то для будь-якої послідовності $s = (b_1, b_2, \dots)$ і будь-якого терма t вираз $s^*(t)$ є елемент множини D , одержаний шляхом підстановки елемента b_i при кожному i замість всіх входжень змінної x_i в терм t і виконання після цього всіх операцій інтерпретації h , які відповідають функціональним символам терма t .

Приклад 3.4.4

Нехай $t = f_2^2(x_4, f_1^2(x_2, a))$, область D — множина цілих чисел \mathbb{Z} , а f_2^2 і f_1^2 інтерпретуються як операції множення та додавання

відповідно і $a = 1$, то для будь-якої послідовності цілих чисел $s = (b_1, b_2, b_3, b_4, \dots)$ значення $s^*(t)$ являє собою ціле число $b_4 \cdot (b_2 + 1)$.

Якщо A є елементарною формулою $A_i^n(t_1, t_2, \dots, t_n)$ і R — відношення, що відповідає цій формулі на D при заданій інтерпретації, то формула A вважається такою, що виконується на послідовності $s = (b_1, b_2, \dots, b_n, \dots)$ тоді і тільки тоді, коли $(s^*(t_1), s^*(t_2), \dots, s^*(t_n)) \in R$.

Формула $\neg A$ виконується на послідовності s тоді і тільки тоді, коли формула A не виконується на s .

Формула $A \rightarrow B$ виконується на s тоді і тільки тоді, коли формула A не виконується на s або коли формула B виконується на s .

Формула $(\forall x_i)(A)$ виконується на $s = (b_1, b_2, \dots)$ тоді і тільки тоді, коли формула A виконується на будь-яких послідовностях із Σ , які відрізняються одна від одної не більше ніж своїми i -ми компонентами.

Формула A називається такою, що виконується, якщо існує інтерпретація h , при якій вона виконується хоча б на одній послідовності інтерпретації h .

Формула A називається **істинною** при заданій інтерпретації h , якщо вона виконується на всякій послідовності із Σ , і **хібною**, якщо вона не виконується на жодній із послідовностей з Σ .

Інтерпретація h називається **моделлю** для заданої множини формул Γ , якщо кожна формула з Γ істинна при інтерпретації h .

Формула A називається **логічно загальнозначимою**, якщо вона істинна при будь-якій інтерпретації.

Формула A називається **суперечністю**, якщо $\neg A$ логічно загальнозначима, або, що те ж саме, A хибна при будь-якій інтерпретації. Говорять, що формула A — **логічний наслідок** множини формул Γ , якщо при деякій інтерпретації формула A виконується на кожній послідовності цієї інтерпретації, на якій виконуються всі формулі з Γ . Формули називаються **логічно еквіва-лентними**, якщо кожна з них є логічним наслідком другої. Розглянемо такі прості твердження, які безпосередньо випливають із наведених вище означень.

Теорема 3.4.1. Для будь-яких формул A і B мають місце такі властивості:

1) A хибна при даній інтерпретації тоді і тільки тоді, коли $\neg A$ істинна в тій же інтерпретації і навпаки;

2) ніяка формула не може бути одночасно хібною і істинною при одній і тій же інтерпретації;

3) якщо в даній інтерпретації істинні формули A і $A \rightarrow B$, то в цій інтерпретації істинна і формула B ;

4) $A \rightarrow B$ хибне при даній інтерпретації тоді і тільки тоді, коли A істинна при цій інтерпретації, а B хибна;

5) $A \& B$ виконується на послідовності s деякої інтерпретації тоді і тільки тоді, коли A виконується на s і B виконується на s . $A \vee B$ виконується на послідовності s тоді і тільки тоді, коли або A виконується на s , або B виконується на s . $A \Leftrightarrow B$ виконується на s тоді і тільки тоді, коли або A виконується на s і B виконується на s , або коли A не виконується на s і B не виконується на s . $\exists x A$ виконується на s тоді і тільки тоді, коли A виконується хоча б на одній послідовності s' , яка відрізняється від послідовності s не більш ніж своїм i -м компонентом;

6) A істинна при даній інтерпретації тоді і тільки тоді, коли $\forall x_i (A)$ істинна при даній інтерпретації;

7) всякий окремий випадок всякої тавтології істинний при будь-якій інтерпретації (окремим випадком тавтології називається фóрмула, одержана шляхом підстановки формул у дану тавтологію замість пропозиційних букв за умови, що замість однієї і тієї ж букви підставляється одна і та ж формула);

8) нехай вільні змінні формули A , якщо вони існують, знаходяться серед змінних x_i, x_j, \dots, x_k . Тоді якщо послідовності s і s' деякої інтерпретації мають одинакові компоненти з номерами i, j, \dots, k , то формула A виконується на послідовності s тоді і тільки тоді, коли вона виконується на послідовності s' ;

9) якщо формула A замкнута, то при будь-якій заданій інтерпретації або істинна A , або істинна $\neg A$;

10) нехай t і v — терми, s — деяка послідовність із Σ , t' одержаний із t шляхом підстановки v замість всіх входжень x_i і s' одержана із s шляхом заміни в ній i -го компонента на $s^*(v)$, тоді $s'(t') = (s')^*(t)$;

11) якщо формула A не включає вільної змінної x_i , то формула $\forall x_i (A \rightarrow B) \rightarrow (A \rightarrow \forall x_i B)$ істинна при будь-якій інтерпретації.

Д о в е д е н н я пропонуються читачеві як вправи.

3. АКСІОМАТИКА І ПРАВИЛА ВИВЕДЕННЯ

Числення предикатів першого порядку \mathbf{K} визначається аксіоматичним шляхом.

Аксіоми поділяються на два класи: логічні аксіоми і власні аксіоми.

Логічні аксіоми: для будь-яких $A, B, C \in \mathbf{K}$, формули
(П1) $A \rightarrow (B \rightarrow A)$;

- (П2) $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C));$
 (П3) $(\neg B \rightarrow \neg A) \rightarrow ((\neg B \rightarrow A) \rightarrow B);$
 (П4) $\forall x_i A(x_i) \rightarrow A(t)$, де $A(x_i)$ є формула із **K** і t є терм із **K**,
 вільний для x_i в $A(x_i)$;
 (П5) $\forall x_i (A \rightarrow B) \rightarrow (A \rightarrow \forall x_i B)$, якщо формула A не включає
 вільних входжень x_i , є аксіомами числення **K**.

Власні аксіоми не можуть бути сформульовані в загальному
 вигляді, оскільки вони змінюються від теорії до теорії.

Правила виведення:

- (i) *modus ponens*: із A і $A \rightarrow B$ випливає B (MP);
 (ii) *правило узагальнення*: із A випливає $\forall x_i A$ (Gen).

Числення предикатів з непустою множиною власних аксіом
 називають **теорією першого порядку**.

Обмеження, наведені у формуллюваннях аксіом, вимагають
 деяких пояснень. Якби в аксіомі (П4) терм t міг бути довільним,
 то стала б можливою така ситуація. Нехай $B(x) = \neg \forall y A(x, y)$ і
 $t = y$. Зауважимо, що t не є вільним для x в $B(x)$. Розглянемо
 такий випадок аксіоми (П4):

$$\forall x (\neg \forall y A_1(x, y)) \rightarrow \neg \forall y A_1(y, y),$$

і візьмемо за область інтерпретації будь-яку область, що має не
 менше двох елементів, а A_1^2 інтерпретується як відношення то-
 тожності. Тоді гіпотеза істинна, а висновок хибний. Отже, і вся
 формула хибна.

Аналогічне справедливе і для аксіоми (П5). Дійсно, нехай A і
 B являють собою $A_1^1(x)$. Таким чином, x входить вільно в A . Роз-
 глянемо окремий випадок аксіоми (П5):

$$\forall x (A_1^1(x) \rightarrow A_1^1(x)) \rightarrow (A_1^1(x) \rightarrow \forall x A_1^1(x)).$$

Очевидно, що гіпотеза в даній формулі істинна. Якщо вибрати
 інтерпретацію так, щоб $A_1^1(x)$ виконувалась не на всіх елементах
 області, а лише на деяких, то виявиться, що висновок у даній
 формулі буде хибним. Отже, і сама формула в даній інтерпретації
 буде хибною.

Моделлю числення **K** називається всяка інтерпретація, при
 якій істинні всі аксіоми теорії **K**.

Легко помітити (див. теорему 3.4.1), що коли правила виведення
 застосовувати до істинних у даній інтерпретації формул, то
 результатом будуть формули, які також істинні в тій же інтерпре-
 тації. Значить, всяка **теорема**, тобто формула, яка виводиться в **K**
 із аксіом, істинна у всякій моделі **K**.

Численні предикатів, крім логічних аксіом, може мати, про що йшла мова вище, і власні аксіоми. Якщо власних аксіом не має, то таке числення називається **численням предикатів першого порядку**.

Числення предикатів з непустою множиною власних аксіом служить для задання теорій і їх моделей для різноманітних алгебр. Як приклад такого числення розглянемо теорію напівгруп.

Приклад 3.4.5 (теорія першого порядку)

Нехай **K** має один бінарний предикатний символ і один бінарний функціональний символ, які запишемо так: $t = s \circ t + s$.

Власними аксіомами теорії **K** є формулі:

- a) $\forall x_1 \forall x_2 \forall x_3 (x_1 + (x_2 + x_3)) = ((x_1 + x_2) + x_3)$ (асоціативність);
- б) $\forall x_1 (x_1 = x_1)$ (рефлексивність);
- в) $\forall x_1 \forall x_2 (x_1 = x_2 \rightarrow x_2 = x_1)$ (симетричність);
- г) $\forall x_1 \forall x_2 \forall x_3 (x_1 = x_2 \rightarrow (x_2 = x_3 \rightarrow x_1 = x_3))$ (транзитивність);
- д) $\forall x_1 \forall x_2 \forall x_3 (x_2 = x_3 \rightarrow (x_1 + x_2 = x_1 + x_3) \wedge$
 $\quad \quad \quad \& (x_2 + x_1 = x_3 + x_1))$ (підстановка).

Всяка модель цієї теорії називається напівгрупою. Якщо в напівгрупі істинна формула $\forall x_1 \forall x_2 (x_1 + x_2 = x_2 + x_1)$, то напівгрупа називається **комутативною**. \blacktriangleleft

4. ОСНОВНІ ВЛАСТИВОСТІ ТЕОРІЙ ПЕРШОГО ПОРЯДКУ

Розглянемо основні властивості логіки предикатів першого порядку. Насамперед установимо такий факт.

Теорема 3.4.2. Всяке числення предикатів першого порядку **K** є несупереченою теорією.

Доведення. Нехай для всякої формули A із **K** $h(A)$ означає формулу, одержану з формули A шляхом вилучення з неї всіх кванторів і термів разом зі всіма відповідними дужками і комами. Наприклад:

$$h(\forall x_2 A_3^2(x_1, x_2) \rightarrow A_1^1(x_1)) = A_3^2 \rightarrow A_1^1;$$

$$h(\neg \forall x_5 A_2^3(x_3, a_1, x_5) \rightarrow A_3^1(x_4)) = \neg A_2^3 \rightarrow A_3^1.$$

Із визначення h випливає, що $h(\neg A) = \neg h(A)$ і $h(A \rightarrow B) = h(A) \rightarrow h(B)$. Звідси маємо, що всяка аксіома (П1)–(П5) перетворюється в тавтологію. Дійсно, це очевидно для (П1)–(П3), а для (П4) маємо

$$h(\forall x_i A(x_i) \rightarrow A(t)) = A \rightarrow A,$$

що теж є тавтологією (як було показано вище). Далі

$$h(\forall x_i (A \rightarrow B) \rightarrow (A \rightarrow \forall x_i B)) = (A \rightarrow B) \rightarrow (A \rightarrow B).$$

Значить, якщо $h(A)$ — тавтологія, то і $h(\forall x_i A)$ — тавтологія. Крім того, як ми впевнилися, коли $h(A)$ і $h(B)$ — тавтології, то і $h(B)$ — тавтологія.

Таким чином, якщо $A \in K$, то $h(A)$ — тавтологія. Якби існувала така формула B , що $B \in K$ і $\neg B \in K$, то формулі B і $\neg B$ одночасно мали б бути тавтологіями, що неможливо внаслідок того, що числення висловлювань є несуперечкою теорією.

Теорема доведена.

Теорема 3.4.3. Якщо $A \in K$ — тавтологія, то A є теоремою в K і може бути виведена в K лише з використанням аксіом (П1)–(П3) і правила MP.

Д о в е д е н н я. Нехай формула A одержана з деякої тавтології B шляхом підстановок. Згідно з теоремою 3.2.6 існує виведення A в численні висловлювань. Зробимо в цьому виведенні всюди підстановки за таким правилом:

(i) якщо яка-небудь пропозиційна буква входить в B , то замість всіх її входжень у кожну з формул виведення підставляємо ту формулу теорії K , яка підставлялась в B ;

(ii) якщо дана пропозиційна буква не входить у B , то замість всіх її входжень у формули виведення підставляємо довільну (але одну і ту ж) формулу теорії K . Одержання таким чином послідовність формул і буде виведенням формули A в теорії K , причому таким, яке використовує лише аксіоми (П1)–(П3).

Теорема доведена.

Означення 3.4.4. Нехай A є формулою деякої сукупності формул Γ і B_1, B_2, \dots, B_n — деяке виведення із Γ з обґрунтуванням кожного кроку в ньому. Будемо говорити, що B_i залежить від A в цьому виведенні, коли:

а) $B_i \in A$ і обґрунтуванням B_i служить належність B_i до Γ ;

б) B_i обґрунтовується як безпосередній наслідок за правилом MP або Gen деяких попередніх в цьому виведенні формул, з яких хоча б одна залежить від A .

П р и к л а д 3.4.6

Розглянемо формулу:

$$A, \forall x A \rightarrow B \vdash \forall x B.$$

- | | | |
|---------------------------------|----------------------|--|
| (1) A | — гіпотеза | (залежить від A); |
| (2) $\forall x A$ | — (1), Gen | (залежить від A); |
| (3) $\forall x A \rightarrow B$ | — гіпотеза | (залежить від A); |
| (4) B | — за MP із (2) і (3) | (залежить від A і $\forall x A \rightarrow B$); |
| (5) $\forall x B$ | — Gen і (4) | (залежить від A і $\forall x A \rightarrow B$). ▲ |

Теорема 3.4.4. Якщо формула B не залежить від формул A при виведенні $\Gamma, A \vdash B$, то $\Gamma \vdash B$.

Доведення індукцією за довжиною виведення $B_1, B_2, \dots, B_n = B$.

При $n = 1$ маємо $B_1 = B$ і B або належить $\Gamma \cup \{A\}$, або є аксіомою. Але B не може збігатися з A внаслідок незалежності B від A . Отже, $\Gamma \vdash B$.

Нехай теорема справедлива для всіх формул B , довжина виведення яких менша n . Якщо $B \in \Gamma$ або B — аксіома, то $\Gamma \vdash B$. Якщо ж B — безпосередній наслідок деяких формул із B_1, B_2, \dots, B_{n-1} (однієї або двох) і оскільки B не залежить від A , то від A не залежить жодна з цих формул. Але тоді, за припущенням індукції, із Γ виводяться ці формули (одна або дві), а разом з ними — і формула B .

Теорема доведена.

Теорема дедукції для числення предикатів не може бути сформульована в такому вигляді, в якому вона формулювалась для числення висловлювань. Але все ж таки деякий варіант цієї теореми існує.

Теорема 3.4.5 (теорема дедукції). Нехай $\Gamma, A \vdash B$ і при цьому нехай існує таке виведення B із Γ, A , в якому жодне застосування правила Gen до формул, які залежать від A в цьому виведенні, не зв'язується квантором загальності формулі A . Тоді $\Gamma \vdash A \rightarrow B$.

Доведення проводиться індукцією за довжиною виведення $B_1, B_2, \dots, B_n = B$.

При $n = 1$ існують дві можливості:

- $B_i \in \Gamma$ або є аксіомою;
- $B_i = A$.

У випадку а) $A \rightarrow B_i$, оскільки це випливає з аксіоми (П1), $B_i \rightarrow (A \rightarrow B_i)$ за правилом МР.

У випадку б), коли $A = B_i$, то $\Gamma \vdash A_i \rightarrow B_i$, оскільки $A \rightarrow A$ — тавтологія (див. приклад 3.2.7).

Нехай тепер теорема справедлива для всіх $i < n$. Припустимо, що існують індекси k і j , менші i , і що $B_k = B_j \rightarrow B_i$. Тоді за припущенням індукції $\Gamma \vdash A \rightarrow B_j$ і $\Gamma \vdash A \rightarrow (B_j \rightarrow B_i)$. Але за аксіомою (П2) маємо

$$A \rightarrow (B_j \rightarrow B_i) \rightarrow ((A \rightarrow B_j) \rightarrow (A \rightarrow B_i))$$

і за правилом МР одержуємо, що $\Gamma \vdash A \rightarrow B_i$.

Припустимо, нарешті, що існує такий індекс $j < i$, що $B_i = \forall x_k B_j$. Тоді за припущенням індукції $\Gamma \vdash A \rightarrow B_i$ і або B_j не залежить від A , або x_k не є вільною змінною формули A . Якщо B_j не залежить від A , то за теоремою 3.4.4 $\Gamma \vdash B_j$ і, застосовуючи

правило Gen, одержуємо $\Gamma \vdash \forall x_k B_j$. За схемою аксіом (П1) заходимо, що

$$B_i \rightarrow (A \rightarrow B_i)$$

i $A \rightarrow B_i$ — за правилом MP.

Якщо x_k не є вільною змінною формули A , то за схемою аксіом (П5) маємо

$$\forall x_k (A \rightarrow B_i) \rightarrow (A \rightarrow \forall x_k B_j).$$

За припущенням індукції $\Gamma \vdash A \rightarrow B_j$, за правилом Gen $\Gamma \vdash \forall x_k (A \rightarrow B_j)$ i, нарешті, за правилом MP одержуємо $\Gamma \vdash A \rightarrow \forall x_k B_j$, тобто $\Gamma \vdash A \rightarrow B_i$.

Теорема доведена.

Наслідок 3.4.1. Якщо $\Gamma, A \vdash B$, i існує виведення В без застосування правила Gen до вільних змінних формули A , то

$$\Gamma \vdash A \rightarrow B.$$

Наслідок 3.4.2. Якщо формула A замкнута i $\Gamma, A \vdash B$, то

$$\Gamma \vdash A \rightarrow B.$$

Приклади

Розглянемо приклади.

1. $\vdash \forall x \forall y A \rightarrow \forall y \forall x A$:

- a) $\forall x \forall y A$ — гіпотеза;
- б) $\forall x \forall y A \rightarrow \forall y A$ — схема аксіом (П4);
- в) $\forall y A$ — MP із а), б);
- г) $\forall y A \rightarrow A$ — схема аксіом (П4);
- д) A — MP із в), г);
- е) $\forall x A$ — Gen із д);
- ж) $\forall y \forall x A$ — Gen із е).

2. $\vdash \forall x (A \rightarrow B) \rightarrow (\forall x A \rightarrow \forall x B)$:

- а) $\forall x (A \rightarrow B)$ — гіпотеза;
- б) $\forall x (A \rightarrow B) \rightarrow (A \rightarrow B)$ — схема аксіом (П4);
- в) $A \rightarrow B$ — MP із а), б);
- г) $\forall x A$ — гіпотеза;
- д) $\forall x A \rightarrow A$ — схема аксіом (П4);
- е) A — MP із г), д);
- ж) B — MP із б), е);
- з) $\forall x B$ — Gen із ж);
- и) $\forall x (A \rightarrow B) \rightarrow (A \rightarrow \forall x B)$ — схема аксіом (П5);
- к) $A \rightarrow \forall x B$ — MP із а), и);

л) $\forall x A \rightarrow A$, $A \rightarrow \forall x B \vdash \forall x A \rightarrow \forall x B$ — на основі транзитивності імплікації.

3. $\vdash \forall x_1 \forall x_2 \dots \forall x_n A \rightarrow A$:

- | | |
|--|----------------------|
| a) $\forall x_1 \forall x_2 \dots \forall x_n A$ | — гіпотеза; |
| б) $\forall x_1 \forall x_2 \dots \forall x_n A \rightarrow \forall x_2 \dots \forall x_n A$ | — схема аксіом (П4); |
| в) $\forall x_2 \dots \forall x_n A$ | — MP із а), б); |
| г) $\forall x_2 \dots \forall x_n A \rightarrow \forall x_3 \dots \forall x_n A$ | — схема аксіом (П4); |
| <hr/> | |
| аа) $\forall x_n A$ | — MP із |
| бб) $\forall x_n A \rightarrow A$ | — схема аксіом (П4); |
| вв) A | — MP із аа), бб). |

Отже, формула A виводиться без застосування правила Gen і за наслідком 3.4.1 остаточно одержуємо

$$\begin{aligned} & \forall x_1 \forall x_2 \dots \forall x_n A \vdash A, \\ & \vdash \forall x_1 \forall x_2 \dots \forall x_n A \rightarrow A. \end{aligned}$$

На прикладі числення висловлювань ми бачили, наскільки корисним є розширення множини правил виведення. Такі правила дають можливість скоротити довжину виведення. Розглянемо деякі з них для числення предикатів.

Правило індивідуалізації (RI). Якщо терм t вільний для змінної x у формулі $A(x)$, то $\forall x A(x) \vdash A(t)$.

Доведення. Для $A(x)$ маємо:

- | | |
|--------------------------------------|----------------------|
| a) $\forall x A(x)$ | — гіпотеза; |
| б) $\forall x A(x) \rightarrow A(t)$ | — схема аксіом (П4); |
| в) $A(t)$ | — MP із а), б). |

Отже, $\forall x A(x) \vdash A(t)$.

Правило існування (RE). Якщо терм t вільний для змінної x у формулі $A(x)$, то $A(t) \vdash \exists x A(x)$.

Доведення. а) $\forall x \neg A(x) \rightarrow \neg A(t)$ — схема аксіом (П4);
 б) $(A \rightarrow \neg B) \rightarrow (B \rightarrow \neg A)$ — тавтологія;
 в) $A(t) \rightarrow \neg \forall x \neg A(x)$ — MP із а), б).

Отже, $A(t) \rightarrow \exists x A(x)$ або $\vdash A(t) \rightarrow \exists x A(x)$.

Правило кон'юнкції (RK). Для будь-яких формул A і B :

$$A, B \vdash A \& B.$$

Доведення. Для A і B маємо:

- | | |
|---|--|
| a) A, B | — гіпотези; |
| б) $A \rightarrow (B \rightarrow \neg(A \rightarrow \neg B))$ | — тавтологія
(див. приклади 3.2.8); |
| в) $B \rightarrow \neg(A \rightarrow \neg B)$ | — MP із а), б); |
| г) $\neg(A \rightarrow \neg B) \Leftrightarrow A \& B$ | — MP із а), в). |

Правило діз'юнкції (RD). Для будь-яких формул A і B

$$A \rightarrow C, B \rightarrow D, \neg A \rightarrow B \vdash \neg C \rightarrow D.$$

Доведення. Для A і B маємо:

- a) $A \rightarrow C, B \rightarrow D, \neg A \rightarrow B, \neg C$ — гіпотези;
- б) $(A \rightarrow C) \rightarrow (\neg C \rightarrow \neg A)$ — тавтологія;
- в) $\neg C \rightarrow \neg A$ — MP із а), б);
- г) $\neg C \rightarrow \neg A, \neg A \rightarrow B \vdash \neg C \rightarrow B$ — транзитивність \rightarrow ;
- д) B — MP із а), г);
- е) $B \rightarrow D$ — гіпотеза;
- ж) D — MP із д), е).

Теорема 3.4.6. Якщо A і B — формули теорії першого порядку K і x — предметна змінна, яка не є вільною змінною у формулі A , то справедливі такі теореми в K :

- (i1) $A \Leftrightarrow \forall x A$;
- (i2) $\exists x A \Leftrightarrow A$;
- (i3) $\forall x (A \rightarrow B) \Leftrightarrow (A \rightarrow \forall x B)$;
- (i4) $\forall x (B \rightarrow A) \Leftrightarrow (\exists x B \rightarrow A)$;
- (i5) $\forall x (A \Leftrightarrow D) \rightarrow (\forall x A \Leftrightarrow \forall x D)$.

Доведення. Для (i1) маємо: а) A — гіпотеза;
б) $\forall x A$ — Gen із а).

Оскільки x не є вільною змінною в A , то $A \vdash \forall x A$ за теоремою дедукції. А це значить, що $\vdash A \rightarrow \forall x A$.

Доведення оберненого (i1) твердження і тверджень (i2)–(i5) пропонуються як вправи.

Нехай змінні x і y не збігаються і формула $A(y)$ одержана з формули $A(x)$ підстановкою у замість всіх вільних входжень x . Тоді формули $A(x)$ і $A(y)$ називаються *подібними*, якщо у вільна для x в $A(x)$ і $A(y)$ не має вільних входжень y . Інакше кажучи, формули $A(x)$ і $A(y)$ подібні тоді і тільки тоді, коли $A(y)$ має вільні входження у точно в тих місцях, в яких $A(x)$ має вільні входження x .

Лема 3.4.1. Якщо формули $A(x)$ і $A(y)$ подібні, то $\vdash \forall x A(x) \Leftrightarrow \forall y A(y)$.

Доведення. За схемою аксіом (П4) маємо $\vdash \forall x A(x) \rightarrow A(y)$. З умови леми і за правилом Gen одержуємо $\vdash \forall y (\forall x (A(x) \rightarrow A(y)))$. За схемою аксіом (П5) маємо $\vdash \forall x A(x) \rightarrow \forall y A(y)$. Аналогічно доводиться і формула $\vdash \forall y A(y) \rightarrow \forall x A(x)$. Скориста-
вшись тавтологією $A \rightarrow (\neg A \rightarrow \neg(A \rightarrow \neg A)) \Leftrightarrow A \rightarrow (B \rightarrow (A \& \& B))$ (теорема з) із §3.2, приклад 3.2.8) і теоремою 3.4.3, одержує-
мо

$$\forall x A(x) \Leftrightarrow \forall y A(y).$$

Лема доведена.

Лема 3.4.2. Якщо замкнута формула $\neg A$ теорії першого порядку K не виводиться в теорії K , то теорія K' , одержана з K внаслідок приєднання формули A як аксіоми, є несуперечною теорією.

Доведення від супротивного. Нехай K' суперечна. Це значить, що існує формула B , для якої має місце $\vdash B$ і $\vdash \neg B$ в теорії K' . Скориставшись тавтологією $\neg A \rightarrow (A \rightarrow B)$ (теорема д із § 3.2, приклад 3.2.8) і теоремою 3.4.3, маємо $\vdash B \rightarrow (\neg B \rightarrow \neg A)$ в теорії K' . Отже, в K' справедливе $\vdash \neg A$, або, що те саме, в теорії K справедливе $A \vdash \neg A$. Оскільки A замкнута, то згідно з наслідком 3.4.2 маємо $\vdash A \rightarrow \neg A$ в K . Але тоді з тавтології $(A \rightarrow \neg A) \rightarrow \neg A$ (теорема б) із § 3.2, приклад 3.2.7) одержуємо, що $\vdash \neg A$ в теорії K . Але останнє суперечить умові леми.

Лема доведена.

Лема 3.4.3. Множина всіх виразів теорії першого порядку K зліченна.

Доведення. Нехай u — деякий символ алфавіту теорії першого порядку. Поставимо у відповідність символу u непарне число $g(u)$ за таким правилом:

$$g(() = 3, g() = 5, g(,) = 7, g(\neg) = 9, g(\rightarrow) = 11, \\ g(x_k) = 5 + 8k, g(a_k) = 7 + 8k, g(f_k^n) = 9 + 8(2^n 3^k), \\ g(A_k^n) = 11 + 8(2^n 3^k).$$

Далі, виразу виду $u_0 u_1 \dots u_r$ поставимо у відповідність число $2^{g(u_0)} 3^{g(u_1)} \dots p_r^{g(u_r)}$, де p_r — r -те просте число. При такій відповідності, очевидно, різні вирази одержують різні номери, і таким чином, можна перелічити всі вирази в порядку зростання чисел, які поставлені їм у відповідність.

Лема доведена.

Теорія першого порядку K називається **повною**, якщо для всякої замкнutoї формулі A теорії K має місце $\vdash A$ або $\vdash \neg A$ в K .

Теорія першого порядку K' , яка має ті ж символи, що й теорія першого порядку K , називається **розширенням теорії K** , коли всяка теорема теорії K є також теоремою теорії K' . Очевидно, що для доведення того, що теорія K' є розширенням теорії K , необхідно довести, що всі власні аксіоми теорії K є теоремами теорії K' .

Лема 3.4.4. Якщо теорія першого порядку несуперечна, то існує несуперечне повне її розширення.

Доведення. Нехай B_1, B_2, \dots — деякий перелік всіх замкнutoих формул теорії першого порядку K (він існує згідно з лемою 3.4.3). Визначимо таку послідовність теорій K_0, K_1, \dots Нехай

хай $K_0 = K$ і припустимо, що K_n визначена. Коли ж невірно, що $\vdash \neg B_{n+1}$ в K_n , то добавляємо формулу B_{n+1} до K_n як нову аксіому і одержуємо теорію K_{n+1} . А якщо $\vdash \neg B$, то покладаємо $K_n = K_{n+1}$. Нехай тепер K' — теорія першого порядку, аксіомами якої є всі аксіоми теорій $K_0, K_1, \dots, K_n, \dots$. Очевидно, що K_{n+1} служить розширенням для K_n , в тому числі і для теорії $K = K_0$. Для доведення несуперечності теорії K' достатньо довести несуперечність кожної з теорій K_i , оскільки всяке виведення суперечності в K' використовує лише скінченне число аксіом, і тому є виведенням суперечності в деякій теорії K_n .

Доведення виконується індукцією за числом n . Для $n = 0$ теорія $K_0 = K$ несуперечна за умовою. Нехай теорія K несуперечна для всіх $i < n + 1$. Тоді якщо $K_{n+1} = K_n$, то і K_{n+1} несуперечна. Якщо ж $K_{n+1} \neq K_n$, то існує формула $\neg B_{n+1}$, яка не виводиться в K_n . Але за лемою 3.4.2 теорія K_{n+1} теж несуперечна. Отже, несуперечна і теорія K' . Покажемо повноту теорії K' .

Нехай A — довільна замкнута формула теорії K' . Ясно, що $A = B_{j+1}$ для деякого $j \in N$. Згідно з визначенням теорії K_j або $\vdash \neg B_{j+1}$ в теорії K_j , або $\vdash B_{j+1}$ в теорії K_{j+1} , оскільки формула B_{j+1} добавляється як аксіома в K_{j+1} . Таким чином, маємо або $\vdash \neg B_{j+1}$, або $\vdash B_{j+1}$ в теорії K_{j+1} .

Лема доведена.

Теорема Геделя. *Всяка несуперечна теорія першого порядку має модель, область якої — зліченна множина.*

Доведення цієї теореми опускається за браком місця, а ознайомитись з її доведенням можна в [32].

Теорія, яка має область інтерпретації зліченну множину, називається *теорією із зліченою моделлю*.

Теорема 3.4.7. *Всяка логічно загальнозначима формула теорії першого порядку K є теоремою теорії K .*

Доведення. Достатньо розглянути лише замкнуті формулі теорії K , оскільки всяка формула B логічно загальнозначима тоді і тільки тоді, коли логічно загальнозначиме її замикання, і B виводиться в K тоді і тільки тоді, коли виводиться її замикання (див. теорему 3.4.1, п. 6).

Нехай A — логічно загальнозначима замкнута формула із K . Якщо A не виводиться в K , то побудуємо теорію K' шляхом додавлення формули $\neg A$ як аксіоми. Теорія K' несуперечна згідно з лемою 3.4.2 і має модель M за теоремою Геделя. Оскільки формула $\neg A$ є аксіомою в K' , то і вона істинна в M . Але внаслідок загальнозначимості A вона теж істинна в M . Отже, формули A і $\neg A$ одночасно істинні в M , чого не може бути згідно з теоремою

3.4.1, п. 2). Таким чином, формула A повинна бути теоремою теорії K .

Теорема доведена.

Теорема про повноту. *Теоремами числення предикатів першого порядку є ті і тільки ті формули, які логічно загальнозначими.*

Д о в е д е н н я. За теоремою 3.4.1, п. 7), аксіоми, що задаються схемами (П1)–(П3), логічно загальнозначими. Згідно з п. 10) і наслідком 3.4.1, а також п. 11), загальнозначимими є і аксіоми, що породжуються схемами (П4) і (П5). Згідно з п. 3) і 4) теореми 3.4.1 правила виведення MP і Gen зберігають властивість формул бути загальнозначимими. Таким чином, всяка теорема будь-якого числення предикатів першого порядку логічно загальнозначима.

Д о в е д е н н я другої половини цієї теореми випливає з теореми 3.4.7.

Теорема доведена.

Наслідок 3.4.3. З теореми повноти випливає:

а) формула A істинна в кожній зліченній моделі теорії K тоді і тільки тоді, коли $\vdash A$ в K , отже, A істинна в кожній моделі теорії K тоді і тільки тоді, коли $\vdash A$ в K ;

б) якщо в кожній моделі теорії K формула B виконується на кожній послідовності, на якій виконуються всі формули деякої множини формул Γ , то $\Gamma \vdash B$ в K ;

в) якщо формула B теорії K є логічним наслідком даної множини формул Γ , то $\Gamma \vdash B$ в K ;

г) якщо формула B теорії K є логічним наслідком формули A тієї ж теорії, то $A \vdash B$ в K .

Д о в е д е н н я пропонуються як вправи.

Наслідок 3.4.4 (теорема Сколема—Левенгейма). Якщо теорія першого порядку K має яку-небудь модель, то вона має і зліченну модель.

Дійсно, якщо K має модель, то K несуперечна (див. теорему 3.4.1, п. 2)). За теоремою Геделя вона має зліченну модель.

Теорема еквівалентності. *Якщо формула B є підформулою формули A і формула A' одержана з формули A заміною яких-небудь (можливо, жодного) входженень B формулою C і якщо всяка змінна формули B або формули C , що є одночасно зв'язаною змінною формули A , зустрічається в списку y_1, y_2, \dots, y_k , то*

$$\vdash \forall y_1 \forall y_2 \dots \forall y_k (B \Leftrightarrow C) \rightarrow (A \Leftrightarrow A').$$

Д о в е д е н н я проводиться індукцією за числом зв'язок і кванторів в A . Насамперед зауважимо, що якщо жодне входження B насправді не заміняється, то A збігається з A' і формула, яку потрібно вивести, є окремим випадком тавтології $B \rightarrow (A \Leftrightarrow A)$.

Якщо B збігається з A і це єдине входження замінюється на C , то формула, яку потрібно вивести, виводиться із аксіоми (П4) (див. теорему 3 прикладів використання теореми дедукції). Отже, надалі можна вважати, що B є власна підформула формулі A і, принаймні, одне входження B замінюється.

Припустимо, що теорема справедлива для всякої формули з меншим числом кванторів і логічних зв'язок, ніж у A .

Випадок 1. A — елементарна формула. Тоді B не може бути власною підформулою формулі A .

Випадок 2. A має вид $\neg D$. Тоді нехай A' являє собою $\neg D'$. За припущенням індукції, $\vdash \forall y_1 \forall y_2 \dots \forall y_k (B \Leftrightarrow C) \rightarrow (D \Leftrightarrow D')$. Звідси за допомогою тавтології $A \Leftrightarrow B \rightarrow (\neg A \rightarrow \neg B)$ одержуємо

$$\vdash \forall y_1 \forall y_2 \dots \forall y_k (B \Leftrightarrow C) \rightarrow (A \Leftrightarrow A').$$

Випадок 3. A має вид $D \rightarrow E$. Нехай $A' = D' \rightarrow E'$. За припущенням індукції, $\vdash \forall y_1 \forall y_2 \dots \forall y_k (B \Leftrightarrow C) \rightarrow (D \Leftrightarrow D')$ і $\vdash \forall y_1 \forall y_2 \dots \forall y_k (B \Leftrightarrow C) \rightarrow (E \Leftrightarrow E')$. Застосовуючи тавтологію

$$((A \Leftrightarrow B) \& (C \Leftrightarrow D)) \rightarrow ((A \rightarrow C) \Leftrightarrow (B \rightarrow D)),$$

одержуємо $\vdash \forall y_1 \forall y_2 \dots \forall y_k (B \Leftrightarrow C) \rightarrow (A \Leftrightarrow A')$.

Випадок 4. A має вид $\forall x D$. Тоді A' являє собою $\forall x D'$. За припущенням індукції, $\vdash \forall y_1 \forall y_2 \dots \forall y_k (B \Leftrightarrow C) \rightarrow (D \Leftrightarrow D')$. Змінна x не є вільною змінною формулі $\forall y_1 \forall y_2 \dots \forall y_k (B \Leftrightarrow C)$. Коли ж допустити, що це так, то x входила б вільно в B або C , і оскільки x зв'язана в A , то x входила б у перелік y_1, y_2, \dots, y_k і була б зв'язаною змінною в $\forall y_1 \forall y_2 \dots \forall y_k (B \Leftrightarrow C)$, а це суперечить умові. Застосовуючи тепер аксіому (П5), одержуємо $\forall y_1 \forall y_2 \dots \forall y_k (B \Leftrightarrow C) \rightarrow (D \Leftrightarrow D')$. За теоремою 3.4.6, (i5) маємо $\forall x (D \Leftrightarrow D') \rightarrow (\forall x D \Leftrightarrow \forall x D')$. Користуючись тепер транзитивністю імплікації, остаточно одержуємо

$$\forall y_1 \forall y_2 \dots \forall y_k (B \Leftrightarrow C) \rightarrow (\forall x D \Leftrightarrow \forall x D')$$

або

$$\forall y_1 \forall y_2 \dots \forall y_k (B \Leftrightarrow C) \rightarrow (A \Leftrightarrow A').$$

Теорема доведена.

Теорема про заміну. Нехай формулі A, A', B і C задовольняють умови теореми еквівалентності. Якщо $\vdash B \Leftrightarrow C$, то $\vdash A \Leftrightarrow A'$, а якщо $\vdash B \Leftrightarrow C$ і $\vdash A$, то $\vdash A'$.

Теорема є простим наслідком теореми еквівалентності.

Теорема про перейменування зв'язаних змінних. Якщо $\forall x B(x)$ є підформулою формулі A , формула $B(y)$ подібна формулі $B(x)$, і A' одержана з A заміною хоча б одного входження $\forall x B(x)$ в A на $\forall y B(y)$, то $\vdash A \Leftrightarrow A'$.

Доведення випливає з леми 3.4.1 і теореми про заміну.

5. НОРМАЛЬНІ ФОРМИ ФОРМУЛ ЛОГІКИ ПРЕДИКАТІВ ПЕРШОГО ПОРЯДКУ

Процедури пошуку доведень теорем у логіці предикатів першого порядку, які будемо розглядати дещо пізніше, застосовуються до деякої “стандартної” форми формул. Розглянемо одну з них, оскільки далі вона часто зустрічатиметься.

Означення 3.4.5. Говорять, що формула Φ логіки предикатів першого порядку знаходиться в *попередній нормальній формі* тоді і тільки тоді, коли вона має вигляд $(Q_1x_1)(Q_2x_2) \dots (Q_nx_n) A$, де (Q_i) є або $(\exists x_i)$ або $(\forall x_i)$ і всі x_i різні для різних i , а A — форма, яка не має кванторів. Приставка $(Q_1x_1) \dots (Q_nx_n)$ називається *префіксом*, а форма A — *матрицею*.

Умовимося, що коли форма A залежить від вільної змінної x , то це буде записуватися як $A(x)$, в протилежному випадку — просто A .

Теорема 3.4.8. Для будь-яких формул F, G, H теорії першого порядку справедливі такі еквівалентності:

- a) $\neg \forall x F(x) \Leftrightarrow \exists x (\neg F(x))$;
- б) $\neg \exists x F(x) \Leftrightarrow \forall x (\neg F(x))$;
- в) $\forall x F(x) \vee G \Leftrightarrow \forall x (F(x) \vee G)$;
 $\forall x F(x) \& G \Leftrightarrow \forall x (F(x) \& G)$;
- г) $\exists x F(x) \vee G \Leftrightarrow \exists x (F(x) \vee G)$;
 $\exists x F(x) \& G \Leftrightarrow \exists x (F(x) \& G)$;
- д) $\forall x F(x) \& \forall x H(x) \Leftrightarrow \forall x (F(x) \& H(x))$;
- е) $\exists x F(x) \vee \exists x H(x) \Leftrightarrow \exists x (F(x) \vee H(x))$;
- ж) $Q_1x F(x) \& Q_2x H(x) \Leftrightarrow Q_1x Q_2y (F(x) \& H(y))$;
- з) $Q_1x F(x) \vee Q_2x H(x) \Leftrightarrow Q_1x Q_2y (F(x) \vee H(y))$.

Доведення. Доведемо а) $\neg \forall x F(x) \Leftrightarrow \exists x (\neg F(x))$. Нехай h — довільна інтерпретація на деякій області D . Якщо $\neg(\forall x F(x))$ істинна в інтерпретації h , то $\forall x F(x)$ хибна в h . Це значить, що існує такий елемент $s^*(x) \in D$, для якого $F(s^*(x))$ хибна, або, те ж саме, що $\neg F(s^*(x))$ істинна в h . Отже, $\exists x (\neg F(x))$ істинна в h .

Далі, якщо $\neg(\forall x F(x))$ хибна в h , то $\forall x F(x)$ істинна в h . Це значить, що $F(x)$ виконується на всіх послідовностях із D або що $\neg F(x)$ не виконується на жодній такій послідовності з D . Отже, $\exists x (\neg F(x))$ хибна в h . Таким чином, $\neg(\forall x F(x)) \Leftrightarrow \exists x (\neg F(x))$.

Доведення б) аналогічне доведенню а).

Тепер доведемо в) $\forall x F(x) \vee G \Leftrightarrow \forall x (F(x) \vee G)$. Нехай маємо внаслідок а) $\forall x F(x) \vee G \Leftrightarrow \neg(\forall x F(x)) \rightarrow G \Leftrightarrow \exists x (\neg F(x)) \rightarrow G$. Отже:

- | | |
|--|------------------------------|
| 1) $\exists x (\neg F(x)) \rightarrow G$
2) $\neg F(x) \rightarrow \exists x (\neg F(x))$ | — гіпотеза;
— правило RE; |
|--|------------------------------|

- 3) $\neg F(x) \rightarrow G$ — транзитивність \rightarrow із 1), 2);
 4) $\forall x (\neg F(x) \rightarrow G) \Leftrightarrow \forall x (F(x) \vee G)$ — Gen із 3).

Навпаки:

- 1) $\forall x (F(x) \vee G) \Leftrightarrow \forall x (\neg F(x) \rightarrow G)$ — гіпотеза;
- 2) $\neg F(x) \rightarrow G$ — правило RI;
- 3) $(\neg F(x) \rightarrow G) \rightarrow (\neg G \rightarrow F(x))$ — тавтологія;
- 4) $\neg G \rightarrow F(x)$ — MP із 2), 3);
- 5) $\forall x (\neg G \rightarrow F(x))$ — Gen із 4);
- 6) $\forall x (\neg G \rightarrow F(x)) \rightarrow (\neg G \rightarrow \forall x F(x))$ — схема аксіом (П5);
- 7) $\neg G \rightarrow \forall x F(x)$ — MP із 5), 6);
- 8) $(\neg G \rightarrow \forall x F(x)) \rightarrow (\neg \forall x F(x) \rightarrow G)$ — тавтологія;
- 9) $\neg \forall x F(x) \rightarrow G \Leftrightarrow \forall x F(x) \vee G$ — MP із 7), 8).

Доведемо спочатку перший із законів г).

- 1) $\exists x F(x) \vee G \Leftrightarrow \forall x (\neg F(x)) \rightarrow G$ — гіпотеза;
- 2) $\neg F(x) \rightarrow G$ — правило RI;
- 3) $(\neg F(x)) \rightarrow G \rightarrow \exists x (\neg F(x) \rightarrow G)$ — правило RE;
- 4) $\exists x (\neg F(x) \vee G) \Leftrightarrow \exists x (F(x) \vee G)$ — MP із 2), 3).

Навпаки:

- 1) $\exists x (\neg F(x) \rightarrow G)$ — гіпотеза;
- 2) $\exists x (\neg F(x) \rightarrow G) \rightarrow (\neg F(x) \rightarrow G)$ — теорема 3.4.6, (i2);
- 3) $\neg F(x) \rightarrow G$ — MP із 1), 2);
- 4) $\forall x (\neg F(x)) \rightarrow \neg F(x)$ — схема аксіом (П4);
- 5) $\forall x (\neg F(x)) \rightarrow G \Leftrightarrow \neg \forall x (\neg F(x)) \vee G \Leftrightarrow \exists x (F(x)) \vee G$ — транзитивність \rightarrow із 3), 4).

Доведення інших законів із в) і г) тепер легко випливають із законів де Моргана. Дійсно,

$$\forall x F(x) \& G \Leftrightarrow \neg(\neg \forall x F(x) \vee \neg G) \Leftrightarrow \neg(\exists x (\neg F(x)) \vee \neg G) \Leftrightarrow \neg(\exists x (\neg F(x) \vee \neg G)) \Leftrightarrow \neg(\exists x \neg(F(x) \& G)) \Leftrightarrow \forall x (F(x) \& G).$$

Аналогічно доводиться і другий закон із г).

Доведення решти еквівалентностей пропонуються як вправи.

Теорема доведена.

З теореми 3.4.8 випливає такий метод побудови ПНФ.

Для того щоб привести формулу Φ до попередньої нормальnoї форми, використовують такі закони:

- i) для вилучення логічних зв'язок $\Leftrightarrow, \rightarrow$:
 - a) $A \Leftrightarrow B = (A \rightarrow B) \& (B \rightarrow A)$,
 - б) $A \rightarrow B = \neg A \vee B$;
- ii) для вилучення і перенесення \neg всередину формул:
 - в) $\neg(\forall x F(x)) = (\exists x)(\neg F(x))$,
 - г) $\neg(\exists x F(x)) = (\forall x)(\neg F(x))$,
 - д) $\neg\neg F = F$;
- iii) для переносу кванторів:

- 1) $Qx F(x) \vee G = Qx (F(x) \vee G);$
- 2) $Qx F(x) \& G = Qx (F(x) \& G);$
- 3) $(\forall x) F(x) \& (\forall x) H(x) = \forall x (F(x) \& H(x));$
- 4) $(\exists x) F(x) \vee (\exists x) H(x) = \exists x (F(x) \vee H(x));$
- 5) $(Q_1x) F(x) \& (Q_2x) H(x) = (Q_1x)(Q_2z) (F(x) \& H(z));$
- 6) $(Q_1x) F(x) \vee (Q_2x) H(x) = (Q_1x)(Q_2z) (F(x) \vee H(z)).$

Приклади 3.4.8

Розглянемо побудову ПНФ:

- 1) $F = (\forall x) P(x) \rightarrow (\exists x) R(x):$

$$(\forall x) P(x) \rightarrow (\exists x) R(x) = \neg((\forall x) P(x)) \vee ((\exists x) R(x)) =$$

$$= (\exists x) \neg P(x) \vee (\exists x) R(x) = (\exists x)(\neg P(x) \vee R(x));$$
- 2) $F = (\forall x)(\forall y)((\exists z) P(x, z) \& P(y, z)) \rightarrow (\exists u) R(x, y, u):$

$$(\forall x)(\forall y)((\exists z) P(x, z) \& P(y, z)) \rightarrow (\exists u) R(x, y, u) =$$

$$= (\forall x)(\forall y)(+(\exists z) P(x, z) \& P(y, z)) \vee (\exists u) R(x, y, u)) =$$

$$= (\forall x)(\forall y)((\forall z)(\neg P(x, z) \vee \neg P(y, z)) \vee (\exists u) R(x, y, u)) =$$

$$= (\forall x)(\forall y)(\forall z)(\exists u)(\neg P(x, z) \vee \neg P(y, z)) \vee R(x, y, u)). \blacktriangleleft$$

Виходячи з наведених міркувань і законів перетворення формул логіки предикатів i)–ii), тепер можна запропонувати алгоритм побудови попередньої нормальної форми.

Алгоритм побудови ПНФ (A)

Початок.

Крок 1. Вилучаємо зв'язки \Leftrightarrow , \rightarrow за допомогою правил а), б).

Крок 2. Переносимо знак \neg всередину формул, користуючись правилами

$$\begin{aligned} \neg(\neg F) &= F, \\ \neg(F \vee G) &= \neg F \& \neg G, \\ \neg(F \& G) &= \neg F \vee \neg G \end{aligned}$$

і правилами в), г).

Крок 3. Перейменовуємо зв'язані змінні, якщо є така необхідність.

Крок 4. Використовуємо правила 1)–6) для одержання попередньої нормальної форми.

Кінець.

Обґрунтуванням даного алгоритму є твердження, яке безпосередньо випливає з теореми 3.4.8.

Теорема 3.4.9. Алгоритм ПНФ перетворює всяку формулу A теорії K в таку формулу B, яка знаходиться в попередній нормальній формі, що $\vdash A \Leftrightarrow B$ в K.

6. СКУЛЕМІВСЬКІ СТАНДАРТНІ ФОРМИ

Вище було розглянуто питання про зведення довільної формули числення предикатів першого порядку до попередньої нормальnoї форми. Оскільки матриця формули не включає кванторів, то її можна подати в кон'юнктивній нормальній формі (КНФ).

Говорять, що формула P знаходиться в кон'юнктивній нормальній формі, якщо вона має вигляд

$$P = P_1 \vee P_2 \vee \dots \vee P_k,$$

де $P_i = A_{i1} \wedge A_{i2} \wedge \dots \wedge A_{in_i}$, і кожне A_{ij} — це атомарна формула або її заперечення.

Нагадаємо процедуру побудови КНФ (а за законом двоїстості і процедуру побудови ДНФ).

Алгоритм побудови КНФ(А)

Початок.

Крок 1. Вилучаємо зв'язки \Leftrightarrow , \rightarrow з формули A за допомогою правил а), б) (як в алгоритмі ПНФ).

Крок 2. Вилучаємо подвійне заперечення за допомогою правила $\neg(\neg F) = F$, і переносимо знак \neg до атомів, користуючись законами де Моргана (як в алгоритмі побудови ПНФ).

Крок 3. Для одержання нормальної форми формули A використовуємо дистрибутивні закони:

$$\begin{aligned} F \vee (G \wedge H) &= (F \vee G) \wedge (F \vee H); \\ F \wedge (G \vee H) &= (F \wedge G) \vee (F \wedge H). \end{aligned}$$

Кінець.

Приклади 3.4.9

Розглянемо:

a) $(P \vee \neg Q) \rightarrow R;$

б) $(P \wedge (Q \rightarrow R)) \rightarrow S;$

Розв'язок:

$$\begin{aligned} \text{а)} \quad (P \vee \neg Q) \rightarrow R &= \neg(P \vee \neg Q) \vee R = (\neg P \wedge \neg(\neg Q)) \vee R = \\ &= (\neg P \wedge Q) \vee R; \\ \text{б)} \quad (P \wedge (Q \rightarrow R)) \rightarrow S &= (P \wedge (\neg Q \vee R)) \rightarrow S = \\ &= \neg(P \wedge (\neg Q \vee R)) \vee S = (\neg P \vee \neg(\neg Q \vee R)) \vee S = \\ &= (\neg P \vee Q) \wedge (\neg P \vee \neg R) \vee S = (\neg P \vee Q \vee S) \wedge (\neg P \vee \neg R \vee S). \end{aligned}$$

Нехай Φ знаходиться в ПНФ $(Q_1x_1) (Q_2x_2) \dots (Q_nx_n) A$, де A має КНФ. Нехай (Q_rx_r) — квантор існування в префіксі $(Q_1x_1) (Q_2x_2) \dots (Q_nx_n)$, $r = 1, 2, \dots, n$. Якщо жодний квантор зага-

льності не стоїть у префіксі лівіше ($Q_r x_r$), то вибираємо константу c , відмінну від інших функціональних символів, які входять до складу формули A , замінююмо всі x_r на c в A і викреслюємо ($Q_r x_r$) з префікса.

Якщо $(Q_{r_1} x_{r_1}), \dots, (Q_{r_k} x_{r_k})$ — список всіх кванторів загальності, які зустрічаються лівіше ($Q_r x_r$) в префіксі, то вибираємо новий k -місний функціональний символ f , відмінний від інших функціональних символів, які входять до складу A , замінююмо всі входження x_r в A на $f(x_{r_1}, \dots, x_{r_k})$ і викреслюємо ($Q_r x_r$) із префікса. Потім застосовуємо цей же процес до всіх кванторів існування в новому префіксі: остання з одержаних формул є *скулемівською стандартною формою*. Константи і функції, які при цьому використовуються для заміни змінних кванторів існування, називаються *скулемівськими функціями*. Нагадаємо деякі означення, наведені в попередньому параграфі.

Диз'юнкт, який складається з k літер, називається *k -літерним диз'юнктом*. Однолітерний диз'юнкт називається *одиничним диз'юнктом*. Коли диз'юнкт не має ніяких літер, то його називають *пустим диз'юнктом*. Оскільки пустий диз'юнкт не має ніяких літер, які могли б виконуватись для будь-якої інтерпретації, то пустий диз'юнкт завжди хибний. Пустий диз'юнкт будемо позначати 0.

Вважається, що множина диз'юнктів S являє собою кон'юнкцію всіх диз'юнктів із S , де кожна змінна в S керована квантором загальності. Завдяки цій домовленості скулемівська стандартна форма може бути просто зображена множиною диз'юнктів.

Теорема 3.4.10 (основна властивість скулемівських стандартних форм). *Нехай S — множина диз'юнктів, які являють собою стандартну скулемовську форму формули Φ . Тоді Φ хибна в тому і тільки в тому випадку, коли хибна S .*

Д о в е д е н н я. Не обмежуючи загальності, будемо вважати, що формула Φ знаходиться в попередній нормальній формі, тобто $\Phi = Q_1 x_1 Q_2 x_2 \dots Q_n x_n M(x_1, x_2, \dots, x_n)$. Нехай Q — перший квантор існування, і $\Phi_1 = \forall x_1 \dots \forall x_{r-1} Q_{r+1} x_{r+1} \dots Q_n x_n M(x_1, x_2, \dots, x_{r-1}, f(x_1, x_2, \dots, x_{r-1}), x_{r+1}, \dots, x_n)$, де f — скулемівська функція, яка відповідає x_r , $r = 1, 2, \dots, n$. Покажемо, що Φ_1 суперечна тоді і тільки тоді, коли Φ суперечна. Припустимо, що Φ — суперечна. Якщо Φ_1 несуперечна, то існує така інтерпретація h , що Φ_1 істинна в h , тобто для всіх x_1, \dots, x_{r-1} існує хоча б один елемент виду $f(x_1, \dots, x_{r-1})$, для якого

$$Q_{r+1} x_{r+1} \dots Q_n x_n M(x_1, \dots, x_{r-1}, f(x_1, \dots, x_{r-1}), x_{r+1}, \dots, x_n)$$

істинна в h . Таким чином, формула Φ істинна в h , а це суперечить припущенняю. Отже, Φ_1 повинна бути суперечністю.

Припустимо тепер, що Φ_1 суперечна. Якщо Φ несуперечна, то існує така інтерпретація h на області D , що Φ істинна в h , тобто для всіх x_1, \dots, x_{r-1} існує такий елемент x_r , що формула $Q_{r+1}x_{r+1} \dots Q_n x_n M(x_1, \dots, x_{r-1}, x_r, x_{r+1}, \dots, x_n)$ істинна в h . Розширимо інтерпретацію h шляхом введення функції $f(x_1, \dots, x_{r-1})$, яка відображає (x_1, \dots, x_{r-1}) на x_r для всіх x_1, \dots, x_{r-1} в D , тобто $f(x_1, \dots, x_{r-1}) = x_r$. Позначимо нову інтерпретацію через h' . Ясно, що для всіх $x_1, \dots, x_{r-1} Q_{r+1}x_{r+1} \dots Q_n x_n M(x_1, \dots, x_{r-1}, f(x_1, \dots, x_{r-1}), x_{r+1}, \dots, x_n)$ істинна в h' , тобто Φ_1 істинна в h' , що суперечить припущення про суперечність Φ_1 . Отже, Φ повинна бути суперечністю.

Нехай Φ має m кванторів існування, $\Phi_0 = \Phi$, і формула Φ_k одержана із формулі Φ_{k-1} заміною першого квантора існування в Φ_{k-1} скулемівською функцією, $k = 1, 2, \dots, m$. Очевидно, що $S = \Phi_m$. Із доведеного вище випливає, що Φ_k суперечна тоді і тільки тоді, коли суперечна формула Φ_{k-1} для $k = 1, 2, \dots, m$. Отже, S суперечна тоді і тільки тоді, коли суперечна Φ .

Теорема доведена.

Слід підкреслити той факт, що одна і та ж формула може мати більше ніж одну стандартну форму. При перетворенні формулі Φ до скулемівської стандартної форми (ССФ) S при виконанні замін кванторів існування скулемівськими функціями останні будуть вибиратися настільки простими, наскільки це можливо.

7. КЛАСИФІКАЦІЯ ЛОГІК

Ознайомившись з численням предикатів першого порядку, або логікою першого порядку, зробимо деякі зауваження відносно виразності цієї логіки.

Слово *перший* у фразі *логіка першого порядку* служить для того, щоб відрізняти логіку цього виду від більш сильних логік: логік другого, третього і т.д. порядків, в яких використовуються деякі позалогічні поняття, такі, як натуральне число чи множина.

Логіка першого порядку — це числення, в якому квантори \forall, \exists завжди діють лише на множині предметних змінних.

Логіка другого порядку дозволяє діяти одному з кванторів на підмножинах (не обов'язково скінченних) множини предметних змінних і на функціональних символах із F . *Слабка логіка другого порядку* дає можливість діяти кванторам на скінченних підмно-

жинах множини предметних змінних і на множині натуральних чисел N .

Логіка третього порядку дозволяє діяти кванторам на множинах функціональних символів із F і т.д.

У цьому ряду логіка першого порядку грає особливу роль, оскільки в ній виражається вся аксіоматика теорії множин. Значить, цечислення займає важливе місце в математиці, для якої теорія множин являє собою фундамент.

Приклади 3.4.10

Розглянемо на конкретних прикладах відомих нам алгебр виражальні можливості логіки першого порядку:

а) якщо алгебра G є групою, то вона служить моделлю для такої системи аксіом:

- (1) $\forall x \forall y \forall z ((x \cdot y) \cdot z = x \cdot (y \cdot z))$,
- (2) $\forall x (x \cdot e = x)$,
- (3) $\forall x \exists y (x \cdot y = e)$

і аксіом б)–д) (див. приклад теорії напівгруп);

б) якщо G — абелева група, то вона додатково ще задовольняє аксіому

- (4) $\forall x \forall y (x + y = y + x)$,

тобто є моделлю для системи аксіом (1)–(4);

в) абелева група, всі елементи якої мають порядки, що не перевищують деякого натурального числа n , служить моделлю для аксіом абелевої групи і аксіоми

- (5) $\forall x (x = 0 \vee 2x = 0 \vee \dots \vee nx = 0)$;

г) абелева група без кручень — це модель для аксіоми

- (6) $\forall n > 1 \forall x (x = 0 \Rightarrow nx = 0)$;

д) абелева періодична група — модель для аксіоми

- (7) $\forall x \exists n > 1 (nx = 0)$.

Аналізуючи наведені приклади, бачимо, що групи а)–в) є моделями логіки першого порядку, оскільки їх аксіоми виражуються в цій логіці.

У випадку абелевих груп без кручень бачимо, що ця група є моделлю слабкої логіки другого порядку, оскільки перший квантор діє на множині N^+ , а не на множині елементів групи G , як у попередніх випадках. Проте аксіому (6) можна замінити нескінченим списком аксіом:

$$\forall x (x = 0 \Rightarrow x = 0),$$

$$\forall x (x \neg= 0 \Rightarrow 2x \neg= 0),$$
$$\forall x (x \neg= 0 \Rightarrow nx \neg= 0),$$

Здебільшого нескінчений список аксіом зручний настільки ж, як і скінчений, але в даному випадку неможливо обйтися скінченим списком аксіом. Це випливає з такого твердження.

Твердження 3.4.1. *Поняття абелевої групи без кручень не є скінченно аксіоматизованим у логіці першого порядку* [40].

Якщо G — абелева періодична група, то це поняття виражається в слабкій логіці другого порядку, але можна показати, що воно не виражається ні скінченим, ні нескінченим числом аксіом у логіці першого порядку. ◀

8. ПОНЯТТЯ ПРО НЕКЛАСИЧНІ ЛОГІКИ

Числення висловлювань і числення предикатів першого порядку називають *арістотелевою* або *класичною логікою*. На відміну від класичної логіки існує цілий ряд інших логік, які називають *некласичними*. Причиною появи некласичних логік є існування великої кількості проблем, для моделювання і розв'язку яких недостатньо формалізму класичної логіки. Некласичні логіки поділяються на два класи: перший клас включає логіки, які розглядаються як розширення класичної логіки, а другий клас — як альтернативи класичній логіці.

До першого класу відносять модальну логіку та її різновиди: *тимпоральну*, *динамічну* та інші, а до другого — *багатозначну*, *часткову*, *нечітку* і *інтуїціоністську логіки*.

Розглянемо коротко деякі з некласичних логік.

8.1. ПРОПОЗИЦІЙНА МОДАЛЬНА ЛОГІКА

8.1.1. Синтаксис

Формули пропозиційної модальної логіки (ПМЛ) будуються за тими ж правилами, що і формули класичної пропозиційної логіки висловлювань. Але на відміну від класичної логіки висловлюваль в ПМЛ використовуються два додаткових символи \Box і \Diamond . Ці символи називаються відповідно *модальними операторами загальності* та *існування*. Обидва оператори унарні і діють на множині формул логіки висловлювань. Таким чином, формули ПМЛ визначаються індуктивно.

Означення 3.4.6 (синтаксис формул ПМЛ).

- Всяка атомарна формула є формуллою ПМЛ.
- Якщо A і B – формули ПМЛ, то $\neg A$, $A \& B$, $A \vee B$, $A \rightarrow B$, $A \leftrightarrow B$, $\Box A$, $\Diamond A$ теж формули ПМЛ.

На оператори \Box і \Diamond накладається співвідношення подвійності: $\Diamond A = \neg \Box \neg A$, яке часто розглядається як означення оператора \Diamond через оператор \Box . Отже, формули ПМЛ завжди можна записувати, використовуючи лише оператор \Box .

Читаються оператори \Box і \Diamond по-різному, і існує кілька варіантів їх читання [67]:

Оператор $\Box A$

Необхідно, щоб A

Завжди буде мати місце A

Вимагається, щоб A

Припускається, що A

Відомо, що A

Всяке виконання

програми дає результат A

Оператор $\Diamond A$

Можливо, що A

Деколи буде мати місце A

Дозволяється, щоб A

Протилежне до A не при-
пускається

Протилежне до A невідомо

Існує таке виконання прог-
рами, яке дає результат A

8.1.2. Семантика

Для визначення семантики формул ПМЛ необхідно розглянути поняття структури.

Означення 3.4.7. Структурою називається пара (W, R) , де W – деяка непуста множина, а $R \subseteq W \times W$ – бінарне відношення на W . Елементи з W називаються *точками*.

Означення 3.4.8. Нехай P – множина атомарних формул ПМЛ. P -моделлю на структурі $F = (W, R)$ називається трійка $M = (W, R, f)$, де $f: P \rightarrow B(W)$ – відображення з множини формул P в булеву множину W . Для $p \in P$ множина $f(p)$ неформально інтерпретується як множина точок із W , в яких формула p істинна.

Якщо P – фіксована множина, то префікс P в означенні моделі опускається, і говорять просто про модель.

Нехай $\omega \in W$ і A – формула ПМЛ. Запис $M \models_{\omega} A$ означає, що формула A істинна в точці ω моделі $M = (W, R, f)$.

Означення 3.4.9 (семантика формул ПМЛ).

- $M \not\models_{\omega} 0$;
- $M \models_{\omega} A$, якщо $\omega \in f(A)$;
- $M \models_{\omega} A \rightarrow B$, якщо з $M \models_{\omega} A$ випливає $M \models_{\omega} B$;

- $M \models_{\omega} \Box A$, якщо з $\omega R t$ випливає $M \models_{\omega} A$ для всіх таких t .

З останнього правила випливає, що формула $\Box A$ істинна у всіх точках t моделі M , які знаходяться у відношенні R з точкою ω . Наведені вище правила називаються **базовими правилами**, за допомогою яких можна обчислювати значення формул 1 , $\neg A$, $A \vee B$, $A \& B$, $A \Leftrightarrow B$ і $\Diamond A$ з використанням еквівалентних перетворень. Наприклад, константу 1 можна записати як $0 \rightarrow A$, а формулу $\neg A$ — як $A \rightarrow 0$ і т. д. З формули $\neg A = A \rightarrow 0$ випливають такі семантичні правила:

- $M \models_{\omega} \neg A$, якщо $\models_{\omega} A$;
- $M \models_{\omega} \Diamond A$, якщо $M \models_t A$ хоча б для одного $t \in W$, такого, що $\omega R t$;
- $M \models_{\omega} A \vee B$, якщо $M \models_{\omega} A$ або $M \models_{\omega} B$;
- $M \models_{\omega} A \& B$, якщо $M \models_{\omega} A$ і $M \models_{\omega} B$.

Означення 3.4.10. Формула A називається *істинною в моделі* M , якщо вона істинна в будь-якій точці цієї моделі, тобто

$$(\forall \omega \in M) M \models_{\omega} A \text{ (позначення } M \models A).$$

Формула A називається *істинною в структурі* $F = (W, R)$, якщо вона істинна в будь-якій моделі $M = (W, R, f)$, тобто

$$(\forall M = (W, R, f)) M \models A.$$

Формула A називається *модальною тавтологією* або *загально-значимою*, якщо вона істинна у всіх структурах (W, R) (позначення $\models A$).

Приклад 3.4.11

1. Показати, що формула $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$ є модальною тавтологією.

Розв'язок. Припустимо, що дана формула хибна. Це можливо лише в тому випадку, коли $\Box(A \rightarrow B)$ істинна, а $(\Box A \rightarrow \rightarrow \Box A)$ хибна, або, що те саме, $\Box(A \rightarrow B)$ істинна, $\Box A$ істинна і $\Box B$ хибна. Істинність формули $\Box A$ означає, що формула A істинна у всіх точках. Істинність формули $\Box(A \rightarrow B)$ означає істинність формули $A \rightarrow B$ у всіх точках. Але з істинності попередньої і одержаної формул випливає істинність формули B у всіх точках, що суперечить тому, що $\Box B$ хибна.

2. Показати, що формула $\Diamond(A \rightarrow B) \rightarrow (\Diamond A \rightarrow \Diamond B)$ є модальною тавтологією.

Розв'язок. Припустимо, що дана формула хибна. Це можливо лише в тому випадку, коли $\Diamond(A \rightarrow B)$ істинна, а $(\Diamond A \rightarrow \rightarrow \Diamond B)$ хибна, або, що те саме, $\Diamond(A \rightarrow B)$ істинна, $\Diamond A$ істинна і $\Diamond B$

хибна. Істинність формул $\Diamond(A \rightarrow B)$ і $\Diamond A$ означає, що $(\forall M = (W, R, f)) \exists t \in W$, таке, що $M \models_t A \rightarrow B$ і $M \models_t A$. Але звідси випливає, що формула B теж істинна в точці t моделі M . Одержана суперечність доводить, що формула $\Diamond(A \rightarrow B) \rightarrow (\Diamond A \rightarrow \Diamond B)$ — модальна тавтологія. ⁴

Розглянуту семантику (формул ПМЛ) називають *реляційною семантикою*, або *семантикою Кріпке*, за ім'ям автора, який вперше її запропонував [67, 77—79].

8.1.3. Аксіоматика ПМЛ

Як зазначалося вище, ПМЛ є розширенням класичної пропозиційної логіки висловлювань. Тому формальна аксіоматична система для цієї логіки будується за тими ж правилами, що і для класичної логіки висловлювань.

Формальна аксіоматична теорія Th для ПМЛ задається таким способом.

1. Символами Th є $\neg, \rightarrow, (,$) і букви A , з алфавіту Al . Символи \neg, \rightarrow називаються *примітивними зв'язками*, а A — *пропозиційними змінними* або *пропозиційними буквами*.

2. Всі букви алфавіту Al є формулами Th . Якщо A і B — формули Th , то $(\neg A), (A \rightarrow B), \Box A$ — формули Th .

3. Для будь-яких формул A, B, C теорії Th формули

$$MA1) A \rightarrow (B \rightarrow A);$$

$$MA2) (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C));$$

$$MA3) (\neg B \rightarrow \neg A) \rightarrow ((\neg B \rightarrow A) \rightarrow B);$$

$$K) \Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$$

є аксіомами (точніше схемами аксіом).

4. Правилами виведення служать:

а) правило *modus ponens* (MP): $A, A \rightarrow B \vdash B$.

б) модальне правило *необхідності* (RN): із $+ \vdash A$ випливає $+ \Box A$.

Інші логічні зв'язки можна ввести за допомогою вже відомих формул:

$$A \& B \Leftrightarrow \neg(A \rightarrow \neg B),$$

$$A \vee B \Leftrightarrow \neg A \rightarrow B,$$

$$A \Leftrightarrow B \Leftrightarrow (A \rightarrow B) \& (B \rightarrow A).$$

Аксіоматика ПМЛ, яка містить аксіому K), називається *формальною модальною логікою*.

Якщо формула A є теоремою деякої модальної логіки Λ , то це записується так:

$$\vdash_{\Lambda} A.$$

Якщо $\{\Lambda_i \mid i \in I\}$ — яка-небудь множина нормальних логік, то їх перетин $\cap \{\Lambda_i \mid i \in I\}$ теж є нормальнюю логікою. Зокрема, логіка K , визначена перетином всіх нормальних логік:

$$K = \cap \{\Lambda_i \mid \Lambda_i \text{ є нормальна логіка}\},$$

є мінімальною модальною логікою, основну властивість якої дає така теорема.

Теорема 3.4.11. *Формула A є теоремою логіки K тоді і тільки тоді, коли A — модальна тавтологія, тобто A істинна у всіх структурах.*

Такі формули

$$\vdash_{\Lambda} A \rightarrow B \Rightarrow \vdash_{\Lambda} \Box A \rightarrow \Box B \text{ і } \vdash_{\Lambda} \Diamond A \rightarrow \Diamond B,$$

$$\vdash_{\Lambda} A \Leftrightarrow B \Rightarrow \vdash_{\Lambda} \Box A \Leftrightarrow \Box B \text{ і } \vdash_{\Lambda} \Diamond A \Leftrightarrow \Diamond B,$$

$$\vdash_{\Lambda} (\Box A \& \Box B) \Leftrightarrow \Box(A \& B),$$

$$\vdash_{\Lambda} \Diamond(A \vee B) \Leftrightarrow (\Diamond A \vee \Diamond B),$$

$$\vdash_{\Lambda} (\Box A \vee \Box B) \rightarrow \Box(A \vee B),$$

$$\vdash_{\Lambda} \Diamond(A \& B) \rightarrow (\Diamond A \& \Diamond B)$$

виконуються в будь-якій нормальній логіці. Вказані вище можливі прочитання модальних формул $\Box A$ і $\Diamond A$ та властивості відношення R складають основу різних застосувань модальної логіки. Деякі з них ґрунтуються на семантиці можливих світів. Суть цієї семантики зводиться до такого.

Непуста множина W , яка входить в означення моделі, називається *універсумом*. Елементи $\omega \in W$, які називались точками, називаються в цій моделі *можливими світами універсуму*. Бінарне відношення R називається *відношенням досяжності*. Таким чином, структура (W, R) складається з можливих світів, які зв'язані між собою відношенням досяжності. Якщо два світи $\omega, \omega' \in W$, то досяжність світу ω' із світу ω позначається $\omega R \omega'$. Розглянемо приклади.

8.1.4. Логіка можливого і необхідного

Ця логіка виникає при конкретизації читання модальних формул $\Box A$ і $\Diamond A$ та властивості відношення R :

- форма $\Box A$ читається так: « A необхідно істинна»;
- форма $\Diamond A$ читається так: «можливо, що A істинна»;
- відношення R рефлексивне, тобто $(\forall \omega \in W) \omega R \omega$,

де необхідною істиною в світі ω називається формула, яка підтверджується у всіх світах, досяжних із ω , а можливою істиною в

світі ω називається формула, яка підтверджується хоча б в одному із світів, досяжних із ω .

Інколи фіксують деякий світ і називають його “реальним світом”. Формула A , яка необхідно істинна в реальному світі, буде істинна у всіх світах, досяжних із реального світу, а згідно з рефлексивністю відношення R і в самому реальному світі. Отже, формула

$$A \rightarrow \Box A$$

істинна в даній структурі.

Якщо формула A можливо істинна в реальному світі, то вона істинна хоча б в одному із світів, досяжних із реального світу. І, зокрема, буде істинна в самому реальному світі, тобто формула

$$A \rightarrow \Diamond A$$

істинна в даній структурі.

8.1.5. Динамічна логіка

Динамічна логіка ґрунтуються на відповідності модальних операторів кожній команді деякої мови програмування. При такій інтерпретації множина W розглядається як множина можливих станів обчислень, а відношення $\omega R\omega'$ свідчать про те, що обчислення починаються в стані ω і закінчуються в стані ω' .

Якщо програма недетермінована (наприклад, включає оператори розпаралелювання), то вона може давати різні результати. Формула $\Box A$ означає в цьому випадку, що всі виконання програми закінчуються тим, що формула A істинна. Формула $\Diamond A$ означає, що хоча б одне виконання програми закінчується тим, що формула A істинна.

Існують і інші модальні логіки, такі, як *логіка віри і знання* [66], *темпоральна логіка* тощо. Темпоральна логіка буде розглянута нижче, оскільки вона має широкі застосування. А зараз розглянемо властивості відношення R і зв'язок цих властивостей з різновидами модальної логіки.

8.1.6. Властивості бінарних відношень і різновиди ПМЛ

Фіксація певних властивостей відношення R дає можливість одержувати різновиди ПМЛ.

Наведений нижче список властивостей відношення R містить як відомі, так і нові властивості.

- | | |
|--|---------------------|
| 1. $\forall s \in W(sRs)$ | — рефлексивність. |
| 2. $\forall s, t \in W(sRt \rightarrow tRs)$ | — симетричність. |
| 3. $\forall s \in W \exists t \in W(Rst)$ | — репродуктивність. |
| 4. $\forall s, t, u \in W(sRt \& tRu \rightarrow sRu)$ | — транзитивність. |

5. $\forall s, t, u \in W(sRt \& sRu \rightarrow tRu)$ — евклідовість.
 6. $\forall s, t, u \in W(sRt \& sRu \rightarrow t = u)$ — часткова функціональність.
 7. $\forall s \in W \exists! t \in W(sRt)$ — функціональність.
 8. $\forall s, t \in W(sRt \rightarrow \exists u(sRu \& uRt))$ — слабка щільність.
 9. $\forall s, t, u \in W\{sRt \& sRu \rightarrow tRu \vee t = u \vee uRt\}$ — слабка зв'язність.
 10. $\forall s, t, u \in W(sRt \& sRu \rightarrow \exists v(tRv \& uRv))$ — слабка направленість.

Запис $\exists! t$ означає: існує один і тільки один елемент t .

Наведеному списку властивостей відповідає список схем формул:

- 1) $\Box A \rightarrow A$ (T);
- 2) $A \rightarrow \Box \Diamond A$ (B);
- 3) $\Box A \rightarrow \Diamond A$ (D);
- 4) $\Box A \rightarrow \Box \Box A$ (4);
- 5) $\Diamond A \rightarrow \Box \Diamond A$ (5);
- 6) $\Diamond A \rightarrow \Box A$;
- 7) $\Diamond A \Leftrightarrow \Box A$;
- 8) $\Box \Box A \rightarrow \Box A$;
- 9) $\Box((A \& \Box A) \rightarrow B) \vee \Box((B \& \Box B) \rightarrow A)$ (L);
- 10) $\Diamond \Box A \rightarrow \Box \Diamond A$.

У правому стовпчику біля схем формул 1)—5), 9) вказані історичні назви відповідних формул. Деякі з наведених схем формул приймаються як схеми аксіом у відповідних аксіоматичних системах. Зв'язок між властивостями 1—10 відношення R і схемами формул 1)—10) дає така теорема.

Теорема 3.4.12. Якщо задана структура $F = (W, R)$, то відношення R тоді і тільки тоді має конкретну властивість 1—10, коли відповідна схема формул істинна в структурі F [52].

Ця теорема пояснює той успіх, який мас реляційна семантика Кріпке. Дійсно, з одного боку, ця семантика добре пристосована для різноманітних застосувань. З другого боку, важливі властивості відношення R , яке входить в означення моделі, підtrzymуються багатьма модальними схемами. Проте існують такі властивості відношення R , які не відповідають загальнозначимості жодної модальної схеми, наприклад:

- | | |
|---|----------------------|
| $\forall s \in W \neg(sRs)$ | — іррефлексивність; |
| $\forall s, t \in W (sRt \& tRs \rightarrow s = t)$ | — антисиметричність; |
| $\forall s, t \in W (sRt \rightarrow \neg(tRs))$ | — асиметричність. |

Прийнято позначати найменшу нормальну логіку, яка включає схеми аксіом $\Sigma_1, \dots, \Sigma_n$, у вигляді $\Lambda = K\Sigma_1 \dots \Sigma_n$. Ця логіка, визначається так:

$$\Lambda = \cap \{\Lambda_i \text{ нормальна і } \Sigma_1 \cap \dots \cap \Sigma_n \subset \Lambda_i\}.$$

Враховуючи ці позначення, для деяких відомих модальних логік застосуємо такі позначення:

$$S4 : KT4, \quad S5 : KT45,$$

де К — найменша нормальна, модальна логіка, а Т, 4 і 5 означає відповідно схему формул Т, 4 і 5, які були наведені вище.

Характеристику деяким таким логікам дають наведені нижче твердження [52].

Теорема 3.4.13. *Формула A є теоремою логіки KT тоді і тільки тоді, коли A істинна в усіх структурах, в яких відношення R рефлексивне.*

Теорема 3.4.14. *Формула A є теоремою логіки S4 тоді і тільки тоді, коли A істинна в усіх структурах, в яких відношення R рефлексивне і транзитивне.*

Теорема 3.4.15. *Формула A є теоремою логіки S5 тоді і тільки тоді, коли A істинна в усіх структурах, в яких відношення R рефлексивне, транзитивне і евклідове (тобто R є відношенням еквівалентності).*

8.2. МУЛЬТИМОДАЛЬНІ ЛОГІКИ

Мультимодальні логіки виникають внаслідок розгляду на множині W не одного відношення R , а сімейства відношень R_i , де $i \in I$, а I — деяка множина індексів. Кожному відношенню R_i ставиться у відповідність модальний оператор загальності $[i]A$. Подвійний до нього модальний оператор існування $\langle i \rangle A$ визначається, як і раніше, $\langle i \rangle A = \neg [i] \neg A$.

Означення 3.4.11. ПМЛ називається пропозиційною мультимодальною логікою (ПММЛ), якщо в ній більше одного модального оператора загальності.

8.2.1. Синтаксис

Синтаксис ПММЛ вводиться індуктивно аналогічно до означення синтаксису формул ПМЛ.

Означення 3.4.12 (синтаксис формул ПММЛ).

- Всяка атомарна формула є формuloю ПММЛ.
- Якщо A і B — формули ПММЛ, то $\neg A$, $A \& B$, $A \vee B$, $A \rightarrow B$, $A \Leftrightarrow B$, $[i]A$, $\langle i \rangle A$ — теж формули ПММЛ.

8.2.2. Семантика

Структура F для мультимодальної логіки визначається так:

$$F = (W, \{R_i \mid i \in I\}),$$

де W — деяка, непуста, множина; $R_i \subseteq W \times W$ — бінарне відношення на W .

Модель $M = (F, f)$ на структурі $F = (W, \{R_i\})$, як і раніше для ПМЛ, визначається за допомогою відображення $f: P \rightarrow \mathbf{B}(W)$, де P — множина атомарних формул ПММЛ, $\mathbf{B}(W)$ — булеан множини W .

Означення 3.4.13 (семантика формул ПММЛ).

- $M \models_{\omega} 0$;
- $M \models_{\omega} A$, якщо $\omega \in f(A)$;
- $M \models_{\omega} A \rightarrow B$, якщо з $M \models_{\omega} A$ випливає $M \models_{\omega} B$;
- $M \models_{\omega} [i]A$, якщо з $\omega R_i t$ випливає $M \models_t A$ для всіх таких $t \in W$.

Означення формули, істинної в моделі M , істинної в структурі F і загальнозначимої, залишаються такими ж, як і для формул ПММЛ.

8.2.3. Аксіоматика ПММЛ

Аксіоми і правила виведення для ПММЛ такі ж, як і для ПМЛ, з тією лише різницею, що схема аксіом **K** і правило модальної необхідності не одні, а для кожного $i \in I$ свої:

$$\text{E}_i) [i](A \rightarrow B) \rightarrow ([i]A \rightarrow [i]B), \\ \text{RMN}_i) \text{ з } \vdash A \text{ випливає } \vdash [i]A,$$

для всіх $i \in I$.

Мультимодальна логіка називається *нормальнюю пропозиційною мультимодальною логікою*, якщо вона включає схеми аксіом **K_i** і правила RMN_i, $i \in I$.

Найменша нормальна логіка позначається **K**.

Контрольні питання

1. Яка різниця між модальною і мультимодальною логіками?
2. Чим відрізняється аксіоматика мультимодальної логіки від аксіоматики модальної логіки?

Задачі і вправи

1. Довести, що формулі

$$\square(A \& B) \Leftrightarrow (\square A \& \square B),$$

$$\Diamond(A \vee B) \Leftrightarrow (\Diamond A \vee \Diamond B),$$

$$\square(A \rightarrow \Diamond(B \rightarrow N)) \rightarrow \Diamond(B \rightarrow (\square A \rightarrow \Diamond B))$$

є модальні тавтології.

2. Показати, що формули $\Box A \rightarrow A$, $\Box A \rightarrow \Box \Box A$, $\Box(A \vee B) \rightarrow (\Box A \vee \Box B)$ істинні не для всіх структур.

8.3. ПРОПОЗИЦІЙНА ТЕМПОРАЛЬНА ЛОГІКА

8.3.1. Синтаксис

Синтаксис пропозиційної темпоральної логіки (ПТЛ) такий, як і синтаксис класичної пропозиційної логіки з тією лише різницею, що в ньому з'являються три нових оператори: \Diamond , \Box і $\Diamond\Box$. Ці оператори називаються **модальними часовими операторами**. Отже, синтаксис ПТЛ визначається як, і для ПМЛ, індуктивно.

Означення 3.4.14 (синтаксис формул ПТЛ).

- Всяка атомарна формула є формuloю ПТЛ.
- Якщо A і B — формули ПТЛ, то $\neg A$, $A \& B$, $A \vee B$, $A \rightarrow B$, $A \Leftrightarrow B$, $\Diamond A$, $\Box A$ — теж формули ПТЛ.

Оператор $\Diamond A$ читається «в наступний момент A », оператор $\Box A$ читається «завжди A » і оператор $\Diamond\Box A$ читається «обов'язково A » або «хоча б раз A ». Ранги цих операторів такі, як і у запереченні.

8.3.2. Семантика

Пару (T, \leq) будемо називати потоком моментів часу або часовою структурою, де T — множина моментів часу, $\leq \subseteq T \times T$ відношення часткового порядку на T , яке часто називають відношенням «раніше—пізніше». Зазначимо, що (T, \leq) — не обов'язково лінійно упорядкована множина, тобто допускаються розгалуження в майбутньому для потоку моментів часу.

Якщо відношення \leq функціональне, то ПТЛ називається **лінійною ПТЛ (ЛПТЛ)**. Нехай h_t — деяка інтерпретація атомарних формул, тобто h_t — відображення, яке надає значення істинності атомарним формулам у момент часу t ($h_t : Al \rightarrow \{0, 1\}$). Відповідно до таблиць істинності можна обчислювати значення $h_t(A)$ формули A в момент часу t . Як же обчислюється значення $h_t(A)$ при заданих значеннях атомарних формул? Припустимо, що в момент часу t можна сказати, що завжди в майбутньому формула A істинна. Для цього і служить оператор \Box , який у формулі $\Box A$ означає, що A має місце в будь-який момент часу s , який більший або дорівнює t в майбутньому. Отже, $h_t(\Box A) = 1$ тоді і тільки тоді, коли для всіх $s \geq t$ маємо $h_s(A) = 1$.

Розширимо визначення функції h_t з атомарних на довільні формулі за допомогою індукції:

$$h_t(\neg A) = 1 - h_t(A);$$

$$h_t(A \& B) = h_t(A) \& h_t(B);$$

- $$h_t(A \vee B) = h_t(A) \vee h_t(B);$$
- $$h_t(A \rightarrow B) = 1 \Leftrightarrow \text{із } h_s(A) = 1 \text{ випливає } h_s(B) = 1;$$
- $$h_t(\Box A) = 1 \Leftrightarrow (\forall s \geq t)(h_s(A) = 1);$$
- $$h_t(\Diamond A) = h_{\text{нп}}(A).$$

Виходячи з цих визначень, знайдемо умови, при яких значення формули $\Box(A \rightarrow B)$ буде істинне. Отже, $h_t(\Box(A \rightarrow B)) = 1 \Leftrightarrow (\forall s \geq t)(h_s(A \rightarrow B) = 1) \Leftrightarrow (\forall s \geq t) \text{ із } h_s(A) = 1 \text{ випливає } h_s(B) = 1.$

Означення 3.4.15. Нехай (T, \leq) — часова структура і $L = \{0, 1\}$ — множина значень істинності. Моделлю формул ПТЛ називається трійка (T, \leq, H) , де H — деяке сімейство функцій $H = \{h_t \mid t \in T\}$, які задають значення атомарних формул для кожного моменту часу t .

Означення 3.4.16. Формула A називається ПТЛ-тавтологією (позначається це $\vdash A$) тоді і тільки тоді, коли $h_t(A) = 1$ для всіх можливих інтерпретацій h_t над довільними можливими моделями часу.

Означення 3.4.17. Формула A називається наслідком формул A_1, A_2, \dots, A_k в ПТЛ (позначення $A_1, A_2, \dots, A_k \vdash A$) тоді і тільки тоді, коли для всіх (T, \leq, H) і всіх $t \in T$ має місце рівність

$$h_t(A_1) + h_t(A_2) + \dots + h_t(A_k) - h_t(A) = k - 1.$$

Наприклад, формула $\Box(A \rightarrow A)$ є як класичною тавтологією (у випадку класичної логіки зв'язка $\Box A \rightarrow A$ збігається з $A \rightarrow A$, так і темпоральною тавтологією. Але формула $A \vee \Box(A \rightarrow A)$ не є темпоральною тавтологією, хоча вона є класичною тавтологією.

Вислів "формула A завжди буде істинною" записують коротко у вигляді формули $\Box A$. А вислів "формула A буде істинною в деякий (невідомий) момент часу в майбутньому" означає те ж саме, що її заперечення не завжди буде істинною формулою, тобто що формула $\neg \Box \neg A$ буде істинна в деякий момент часу.

8.3.3. Аксіоматика ЛПТЛ

Для будь-яких формул A, B, C ЛПТЛ формули

- A1) $A \rightarrow (B \rightarrow A);$
- A2) $((A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C));$
- A3) $(\neg B \rightarrow \neg A) \rightarrow ((\neg B \rightarrow A) \rightarrow B);$
- K) $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B);$
- T1) $\Diamond A \Leftrightarrow \neg \Box \neg A;$
- T2) $\Diamond(A \rightarrow B) \rightarrow (\Diamond A \rightarrow \Diamond B);$
- T3) $\Box A \rightarrow (A \wedge \Diamond A \wedge \Box \Diamond A);$
- T4) $\Box(A \rightarrow \Diamond A) \rightarrow (A \rightarrow \Box A);$

T5) $OA \Leftrightarrow \neg O\neg A$

є аксіомами (точніше, схемами аксіом).

Правилами виведення служать:

а) правило *modus ponens* (MP): $A, A \rightarrow B \vdash B;$

б) *модальне правило необхідності* (RN): із $\vdash A$ випливає $\vdash \Box A$.

Наведена аксіоматика є повною і несуперечною. Зв'язок між синтаксисом і семантикою формул ЛПТЛ дає таке твердження [52].

Теорема 3.4.16. Формула A є теоремою ЛПТЛ тоді і тільки тоді, коли A — тавтологія ЛПТЛ.

Розглянемо деякі властивості ЛПТЛ.

Теорема 3.4.17. Для будь-якої формули A справедливі такі твердження:

- (а) $\Box A \rightarrow A$; (б) $\Box A \rightarrow OA$; (в) $A \rightarrow \Diamond A$;
(г) $OA \rightarrow \Diamond A$; (д) $\Box A \rightarrow \Diamond A$; (е) $\neg OA \Leftrightarrow O\neg A$.

Д о в е д е н н я. Покажемо спочатку справедливість такого майже очевидного твердження:

$$A \wedge B \rightarrow A.$$

1) $\neg A \rightarrow (B \rightarrow \neg A)$ — аксіома A1);

2) $\neg(B \rightarrow \neg A) \rightarrow A$ — тавтологія;

3) $\neg(B \rightarrow \neg A) \rightarrow A = A \wedge B \rightarrow A$ — еквівалентні перетворення.

Тепер доведемо теорему:

а) із аксіоми T3), транзитивності імплікації і доведеного вище твердження маємо: із $\Box A \rightarrow (A \wedge OA \wedge O\Box A)$ і $(A \wedge OA \wedge O\Box A) \rightarrow A$ випливає $\Box A \rightarrow A$;

б) доводиться аналогічно пункту а);

в) 1) $\Box \neg A \rightarrow \neg A$ — теорема пункта а),

2) $\neg(\neg A) \rightarrow \neg \Box \neg A$ — тавтологія,

3) $A \rightarrow \Diamond A$ — аксіома T1);

г) 1) $\Box \neg A \rightarrow O \neg A$ теорема пункта б),

2) $\neg O \neg A \rightarrow \neg \Box \neg A$ — тавтологія,

3) $OA \rightarrow \Diamond A$;

д) 1) $\Box A \rightarrow A$ — теорема пункта а),

2) $A \rightarrow \Diamond A$ — теорема пункта в),

3) $\Box A \rightarrow \Diamond A$ — транзитивність імплікації;

е) 1) $O \neg A \Leftrightarrow \neg O \neg \neg A$ — аксіома T5),

2) $O \neg A \Leftrightarrow \neg OA$ — закон подвійного заперечення.

Теорема доведена.

Аксіома T4) в наведеній вище системі аксіом є не що інше як аксіома індукції. Дійсно, індуктивний крок доведення за індукцією є $A \rightarrow OA$, тобто припускаючи, що A вірно «сьогодні», дово-

димо, що A вірно «завтра». Якщо індуктивний крок завжди існує $\Box(A \rightarrow \Box A)$, то звідси виводиться, що $A \rightarrow \Box A$. Доведемо це і ще деякі інші правила строго.

Теорема 3.4.18. Для будь-яких формул A і B справедливі такі твердження:

- a) $(A \rightarrow \Box A) \rightarrow \vdash (A \rightarrow \Box A)$; б) $(A \rightarrow B) \rightarrow \vdash (\Box A \rightarrow \Box B)$;
- в) $(A \rightarrow B) \vdash (\Box A \rightarrow \Box B)$.

Д о в е д е н н я.

- | | | |
|----|--|-------------------------|
| a) | 1) $A \rightarrow \Box A$ | — гіпотеза, |
| | 2) $\Box(A \rightarrow \Box A)$ | — правило RN, |
| | 3) $\Box(A \rightarrow \Box A) \rightarrow (A \rightarrow \Box A)$ | — аксіома T4), |
| | 4) $A \rightarrow \Box A$ | — правило MP із 2), 3); |
| б) | 1) $A \rightarrow B$ | — гіпотеза, |
| | 2) $\Box(A \rightarrow B)$ | — правило RN, |
| | 3) $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$ | — аксіома K), |
| | 4) $\Box A \rightarrow \Box B$ | — правило MP із 2), 3); |
| в) | 1) $A \rightarrow B$ | — гіпотеза, |
| | 2) $\Box(A \rightarrow B)$ | — правило RN, |
| | 3) $\Box(A \rightarrow B) \rightarrow \Box(A \rightarrow B)$ — пункт б) попередньої теореми, | |
| | 4) $\Box(A \rightarrow B)$ | — правило MP із 2), 3), |
| | 5) $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$ — аксіома T2), | |
| | 6) $\Box A \rightarrow \Box B$ | — правило MP із 4), 5). |

Теорема доведена.

Контрольні питання

1. Дайте визначення синтаксису і семантики ПТЛ.
2. Коли ПТЛ називається лінійною?
3. Дайте словесну інтерпретацію аксіом T1) – T5).

Задачі і вправи

1. Довести теореми в ЛПТЛ:
 - $\Box A \Leftrightarrow (A \wedge \Box \Box A)$;
 - $\Diamond A \Leftrightarrow (A \vee \Box \Box A)$.

8.4. ПРОПОЗИЦІЙНА n -ЗНАЧНА ЛОГІКА

Алфавіт, правила побудови формул і аксіоматика для цієї логіки такі ж, як і для класичної пропозиційної логіки висловлювань. Різниця полягає в тому, що множина логічних значень

L_n складається не з двох значень: 0 і 1, як в численні висловлювань, а з n значень — 0, $1/(n-1)$, $2/(n-1)$, ..., $(n-2)/(n-1)$, 1, тобто $L_n = \{0, 1/(n-1), \dots, (n-2)/(n-1), 1\}$, де 0 і 1, як і в численні висловлювань, являють собою хибність і істину відповідно, а решта значень — степінь істинності між 0 і 1. Зокрема, при $n = 2$ маємо 2-значну логіку ($L_2 = \{0, 1\}$), при $n = 3$ маємо 3-значну логіку ($L_3 = \{0, 1/2, 1\}$) і т. д.

Розглянемо модельні (семантичні) аспекти n -значної логіки на прикладі 3-значної логіки, оскільки ці аспекти характерні для логік і при $n > 3$.

Означення 3.4.18. Відображення $h: Al \rightarrow L_n$ називається інтерпретацією, де Al — алфавіт атомарних формул пропозиційної n -значної логіки, а $L_n = \{0, 1/(n-1), \dots, (n-2)/(n-1), 1\}$.

При заданій інтерпретації можна обчислювати значення формул. Обчислення значень формул n -значної логіки, як і формул 2-значної логіки, виконуються за допомогою таблиць істинності. Ці таблиці задають загальний спосіб обчислення значень формул для будь-якого n .

A	B	$A \& B$	$A \vee B$	$A \rightarrow B$	$\neg A$
x	y	$\min(x, y)$	$\max(x, y)$	$\min(1, 1 - x + y)$	$1 - x$
1	1	1	1	1	0
1	$1/2$	$1/2$	1	$1/2$	0
1	0	0	1	0	0
$1/2$	1	$1/2$	1	1	$1/2$
$1/2$	$1/2$	$1/2$	$1/2$	1	$1/2$
$1/2$	0	0	$1/2$	$1/2$	$1/2$
0	1	0	1	1	1
0	$1/2$	0	$1/2$	1	1
0	0	0	0	1	1

В даній таблиці пояснення вимагає лише обчислення значення формули $A \rightarrow B$, оскільки решта формул обчислюється традиційно. Формула $A \rightarrow B$, як і в численні висловлювань, вважається істинною, якщо формула B не менш істинна, ніж формула A , тобто значення формули B не менше значення формули A . Це узгоджується з тим, що формулі $0 \rightarrow 0$, $0 \rightarrow 1$, $1 \rightarrow 1$ істинні, як і у випадку 2-значної логіки. Якщо x — значення формули A , а y — значення формули B , то різниця $x - y$ показує, наскільки значення формули $A \rightarrow B$ відхиляється від істини. Отже, формулі $A \rightarrow B$ можна поставити у відповідність значення $1 - (x - y) = 1 - x + y$. Комбінуючи обидва можливі випадки $x \leq y$ і $x > y$, формула $A \rightarrow B$ співставляється зі значенням $\min(1, 1 - x + y)$.

З даної таблиці очевидним чином випливає справедливість таких законів:

- 1) $A \& B = B \& A$, $A \vee B = B \vee A$;
- 2) $A \& (B \& C) = (A \& B) \& C$, $A \vee (B \vee C) = (A \vee B) \vee C$;
- 3) $\neg\neg A = A$;
- 4) $A \& 0 = 0$, $A \& 1 = A$;
- 5) $A \vee 0 = A$, $A \vee 1 = 1$;
- 6) $\neg(A \& B) = (\neg A \vee \neg B)$.

Означення 3.4.19. Формула A пропозиційної n -значної логіки називається L_n -тавтологією (L_n -суперечністю), якщо вона приймає значення 1 (0) при будь-якій інтерпретації h .

Приклад 3.4.12

1. Показати, що формула $A \rightarrow A \in L_3$ -тавтологією.

Розв'язок. Для формули $A \rightarrow A$

- a1) при $h(A) = 0$ маємо $\min(1, 1 - 0 + 0) = \min(1, 1) = 1$;
- a2) при $h(A) = 1/2$ маємо $\min(1, 1 - 1/2 + 1/2) = \min(1, 1) = 1$;
- a3) при $h(A) = 1$ маємо $\min(1, 1 - 1 + 1) = \min(1, 1) = 1$.

Отже, при будь-якій інтерпретації формула $A \rightarrow A$ приймає значення "істина", а це значить, що вона є L_3 -тавтологією.

2. Показати, що формула, $\neg A \vee A$ не є L_3 -тавтологією.

Розв'язок. Для формули $\neg A \vee A$:

- b1) при $h(A) = 0$ маємо $\max(1, 0) = 1$;
- b2) при $h(A) = 1/2$ маємо $\max(1/2, 1/2) = 1/2$;
- b3) при $h(A) = 1$ маємо $\max(0, 1) = 1$.

Отже, оскільки при інтерпретації б2 формула не є істиною, то вона не є і L_3 -тавтологією. \square

Останній приклад показує, що закон «виключення третього» не справедливий в n -значній логіці ($n \geq 3$). А це значить, що така логіка суттєво відрізняється від 2-значної. Крім того, цей факт свідчить, що формули $A \rightarrow B$ і $\neg A \vee B$ нееквівалентні між собою в цій логіці.

Нехай $A(A_1, A_2, \dots, A_k)$ означає, що формула A залежить від атомів A_1, A_2, \dots, A_k або що формула A побудована з цих атомів за допомогою логічних зв'язок. Оскільки всяка формула n -значної логіки A , яка залежить від атомарних формул A_1, A_2, \dots, A_k , може мати n^k інтерпретацій, то має місце таке твердження.

Твердження 3.4.2. Число всіх формул n -значної логіки, які залежать від атомарних формул A_1, A_2, \dots, A_k , дорівнює n^{n^k} .

Часто разом з n -значною логікою для скінченного n розглядають і ∞ -логіку, де L_∞ — множина раціональних чисел x , таких,

що $0 \leq x \leq 1$. Обчислення значень формул ∞ -значеній логіки ведеться за правилами, аналогічними тим, що описані в наведений вище таблиці істинності. Наприклад, користуючись цими таблицями, легко переконатися, що формули є L_∞ -тавтологіями.

- $A \rightarrow A$;
- $A \rightarrow (B \rightarrow A)$;
- $(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$;
- $(A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C))$;
- $((A \rightarrow B) \rightarrow B) \rightarrow ((B \rightarrow A) \rightarrow A)$;
- $(A \rightarrow B) \rightarrow ((B \rightarrow A) \rightarrow (B \rightarrow A))$.

Означення 3.4.20. Формула B називається наслідком формул A_1, A_2, \dots, A_k в n -значній чи ∞ -значній логіці (позначення $A_1, A_2, \dots, A_k, A_k \vdash_n B$) тоді і тільки тоді, коли для довільної інтерпретації h має місце нерівність

$$h(A_1) + h(A_2) + \dots + h(A_k) - h(B) \leq k - 1.$$

Зокрема, $\vdash_n B$ тоді і тільки тоді, коли $h(B) = 1$.

Теорема 3.4.19 (теорема дедукції). $A_1, A_2, \dots, A_k, B \vdash_n C$ тоді і тільки тоді, коли $A_1, A_2, \dots, A_k \vdash_n B \rightarrow C$.

Існують і інші визначення того, що формула B є логічним наслідком формул A_1, A_2, \dots, A_k в n -значній логіці. Наприклад,

$$A_1, A_2, \dots, A_k \vdash_n B \Leftrightarrow \forall h \min\{h(A_i)\} \leq h(B).$$

Контрольні питання

- Чим відрізняється 3-значна логіка від 2-значної?
- Чи всі закони 2-значної логіки виконуються в n -значній логіці?
- Скільки існує формул 4-значної логіки, які залежать від атомарних формул A, B, C ?

Задачі і вправи

- Довести закони 1)–6) для n -значної логіки.
- Довести, що формули 1)–6) є L_∞ -тавтологіями.
- Обчислити значення 3-значної логіки:
 - $\neg A \vee B$,
 - $\neg A \& \neg B$,
 - $A \& \neg A$,
 - $((A \& B) \rightarrow C) \rightarrow (A \rightarrow (B \rightarrow C))$,
 - $(A \rightarrow B) \& (B \& A)$,
 - $(A \rightarrow (B \& C)) \rightarrow ((A \rightarrow B) \& (A \rightarrow C))$.

4. Побудувати формули 3-значної логіки $B(A)$, $C(A)$, $D(A)$, які залежать від атомарної формули A , такі, що $B(A) \in L_3$ -тавтологією, $C(A)$ приймає значення $1/2$ і $D(A) \in L_3$ -суперечністю.

5. Показати, що $((A^n \rightarrow B) \rightarrow A) \rightarrow A \in L_n$ -тавтологією, але не є L^{n+1} -тавтологією, де $A^n \rightarrow B$ визначається рекурсивно:

$$A^0 \rightarrow B \in B, A^{n+1} \rightarrow B \in A \rightarrow (A^n \rightarrow B).$$

6. Довести теорему дедукції для n -значної пропозиційної логіки.

9. ПОНЯТТЯ АЛГЕБРАЇЧНОЇ СИСТЕМИ

Поняття алгебраїчної системи поєднує в собі поняття універсальної алгебри і логічного числення. Формальним апаратом теорії алгебраїчних систем є числення предикатів першого порядку, а сама теорія може розглядатися як теорія, що знаходиться між математичною логікою і алгеброю.

Розглянемо коротко поняття алгебраїчної системи і найпростіші властивості цього поняття.

Алгебраїчною системою (AC) називається об'єкт $A = (A, \Omega, \Pi)$, який складається з трьох множин: непустої множини A , множини операцій $\Omega = \{\omega_1^n, \omega_2^n, \dots\}$ (сигнатура операцій), що визначені на множині A , і множини предикатів $\Pi = \{A_1^k, A_2^k, \dots\}$ (сигнатура предикатів), які задані на множині A . Типом AC називається пара множин $\tau = (\{n_1, n_2, \dots\}, \{k_1, k_2, \dots\})$, елементами яких слугують відповідно арності операцій і предикатних символів. Тип AC τ скінчений, якщо скінченні обидві його множини. Множина A , як і у випадку універсальних алгебр, називається *носієм* або *основною множиною системи* $A = (A, \Omega, \Pi)$, а її елементи — *елементами системи A*. Потужність $|A|$ множини A називається *потужністю*, або *порядком AC* $A = (A, \Omega, \Pi)$. На відміну від інших операцій і предикатів, які можуть бути визначені на множині A , операції з Ω і предикати з Π називаються *основними*. Значення основних нульлярних операцій AC називаються *головними* або *виділеними елементами* цієї системи.

Алгебраїчні системи $A = (A, \Omega, \Pi)$ і $B = (B, \Omega', \Pi')$ називаються *алгебраїчними системами одного і того ж типу*, якщо між множинами їх операцій і предикатів існує така взаємно однозначна відповідність, при якій відповідні операції і предикати мають одинакові арності. Відповідні операції називають операціями *одного і того ж типу* або *однотипними*. Часто, як і у випадку універсальних алгебр, однотипні операції і предикати не розрізняють між собою і вважають, що системи $A = (A, \Omega, \Pi)$ і $B = (B, \Omega', \Pi')$ мають одні і ті ж сигнатури операцій і предикатів.

AC скінченного типу $A = (A, \Omega, \Pi)$ називається *скінченою*, якщо множина A скінчена. У цьому випадку AC $A = (A, \Omega, \Pi)$ записується $A = (A; \omega_1^n, \dots, \omega_s^n; A_1^k, \dots, A_r^k)$.

AC $A = (A, \Omega, \Pi)$ називається *алгеброю*, якщо $\Pi = \emptyset$, і *моделлю*, якщо $\Omega = \emptyset$.

Приклади 3.4.13

$$\begin{aligned} A &= (\mathbf{Z}; \{+\}), \\ A &= (\mathbf{RC}; \{+, -\}), \\ A &= (\mathbf{Z}; \{+\}; \{\leq\}), \\ A &= (\mathbf{Z}; \{\leq\}), \\ A &= (\mathbf{N}; \{s, +, 0, 1\}), \end{aligned}$$

де \mathbf{Z} , \mathbf{RC} , \mathbf{N} — множини всіх цілих, раціональних і натуральних чисел відповідно, а “ $+$ ” і “ $-$ ” — звичайні операції додавання і віднімання чисел. Предикат \leq — це відношення лінійного порядку. Перша і друга AC — алгебри типу (2) і (2, 2), третя AC — типу ((2); (2)), четверта — модель типу (2), а п'ята — алгебра типу (1, 2, 0, 0), де $s(x) = x + 1$, а 0 і 1 — головні елементи цієї AC. *

Якщо у визначенні AC замінити слово *операції* словами *часткових операцій*, то одержимо визначення *часткової AC*.

Як нам відомо з розділу 1 (див. § 1.1, п.4.9), всяку n -арну операцію можна розглядати як $(n + 1)$ -арне відношення. Нехай A_w означає предикат, який відповідає відношенню, що задається n -арною операцією ω , тобто

$$A_w(a_1, a_2, \dots, a_n, b) = 1 \Leftrightarrow \omega(a_1, a_2, \dots, a_n) = b,$$

де $a_1, a_2, \dots, a_n, b \in A$.

Якщо таким чином замінити в AC всі операції предикатами, то одержимо модель $A^* = (A, \Pi^*)$, яку називають *моделлю*, що представляє AC $A = (A, \Omega, \Pi)$.

Приклад 3.4.14

Якщо $A = (\mathbf{Z}; \{+, -, 0\}; \{\leq\})$, то $A^* = (\mathbf{Z}; \{A_+^2, A_-^2, A_0^1, \leq\})$, де

$$\begin{aligned} A_+(x, y, z) &= 1 \Leftrightarrow x + y = z, \\ A_-(x, y, z) &= 1 \Leftrightarrow x - y = z, \\ A_0(x) &= 1 \Leftrightarrow x = 0. \end{aligned}$$

Відображенням AC $A = (A, \Omega, \Pi)$ в AC $B = (B, \Omega', \Pi')$ називається відображення $h: A \rightarrow B$.

Ізоморфізмом АС $A = (A, \Omega, \Pi)$ типу τ в АС $B = (B, \Omega', \Pi')$ того ж типу τ називається взаємно однозначне відображення h , яке задовільняє такі умови:

$$h(\omega(a_1, a_2, \dots, a_n)) = \omega'(h(a_1), h(a_2), \dots, h(a_n)), \quad (\text{i})$$

$$A_k(a_1, a_2, \dots, a_n) = A'_k(h(a_1), h(a_2), \dots, h(a_n)), \quad (\text{ii})$$

для будь-яких a_1, a_2, \dots, a_n із A і для відповідних ω з Ω і ω' з Ω' , A_k з Π і A'_k з Π' . Ізоморфізм АС $A = (A, \Omega, \Pi)$ на себе називається **автоморфізмом**.

Гомоморфізмом $A = (A, \Omega, \Pi)$ в АС $B = (B, \Omega', \Pi')$ того ж типу, що й A , називається відображення АС A в АС B , яке задовільняє умову (i) і умову

$$A_k(a_1, a_2, \dots, a_n) \rightarrow A'_k(h(a_1), h(a_2), \dots, h(a_n)) \quad (\text{i2})$$

для всіх $a_1, a_2, \dots, a_n \in A$, A_k з Π і A'_k з Π' .

З цих означенень випливає, що всякий ізоморфізм АС є гомоморфізмом. Взаємно однозначний гомоморфізм для АС не завжди буде ізоморфізмом. Дійсно, нехай $A = (N, A)$ і $B = (N, <)$, де A — бінарне тотожно хибне відношення на N . Моделі A і B одного типу, а довільне відображення A в B буде гомоморфізмом. Справді, умова (i2) тут завжди виконується, а умову (i) і перевіряти не потрібно, оскільки немає операцій. Але між A і B немає ніякого ізоморфізму. Якби це було не так, тобто якщо h — ізоморфізм, то $h(0)$ і $h(1)$ різні і, значить, або $h(0) < h(1)$, або $h(1) < h(0)$. Із умови (ii) випливає, що $A(0, 1)$ або $A(1, 0)$ буде істинним, що суперечить тому, що A тотожно хибне на N .

Гомоморфізм $h: A = (A, \Omega, \Pi)$ в АС $B = (B, \Omega', \Pi')$ називається **гомоморфізмом “на”**, якщо h — відображення A на B .

Безпосередньо з означень гомоморфізму та ізоморфізму випливає така теорема.

Теорема 3.4.20. 1. *Всякий гомоморфізм h скінченної АС $A = (A, \Omega, \Pi)$ на себе є ізоморфізмом.*

2. *Добуток гомоморфізмів (ізоморфізмів) АС є гомоморфізмом (ізоморфізмом) АС.*

3. *Відношення ізоморфізму АС є відношенням еквівалентності.*

Доведення пропонуються як прості вправи.

Теорема 3.4.21. *Відображення $h: A = (A, \Omega, \Pi)$ в АС $B = (B, \Omega', \Pi')$ є гомоморфізмом тоді і тільки тоді, коли h — гомоморфізм моделі A^* в модель B^* , де A^* і B^* — моделі, які являють собою відповідні АС A і B .*

Доведення. Нехай h — гомоморфізм A^* в B^* , а ω і ω' — одноіменні основні операції АС A і B , а A_w і $A'_{w'}$ — відповідні

предикати моделей A^*, B^* . Нехай $a_1, \dots, a_n \in A$. Тоді покладемо $\omega(a_1, \dots, a_n) = a$. Одержано $A_w(a_1, \dots, a_n, a) = 1$. Значить, $A_w(h(a_1), \dots, h(a_n), h(a)) = 1$, тобто $\omega'(h(a_1), \dots, h(a_n)) = h(a)$.

Доведення у зворотному порядку проводиться аналогічно.

Теорема доведена.

АС $A_1 = (A_1, \Omega, \Pi)$ називається підсистемою АС $A = (A, \Omega, \Pi)$, якщо $A_1 \subseteq A$ і множина A_1 замкнута відносно будь-якої основної операції з Ω , а значення основних предикатів із Π на множині A_1 збігаються із значеннями тих же предикатів на A для відповідних елементів. Якщо A — алгебра або модель, то A_1 називається *підалгеброю* або *підмоделлю* A . Зокрема, якщо A являє собою модель, то довільна непуста підмножина носія буде замкнутою і, значить, буде підмоделлю моделі A . Як і для універсальних алгебр, так і для АС, має місце така теорема.

Теорема 3.4.22. *Перетин довільної сукупності підсистем АС $A = (A, \Omega, \Pi)$, якщо він непустий, буде підсистемою АС A .*

Контрольні питання

1. Дайте означення логічно загальнозначимої формули.
2. Чим відрізняються формули числення висловлювань від формул числення предикатів?
3. Скільки аксіом і правил виведення має теорія першого порядку?
4. Скільки аксіом і правил виведення має числення предикатів першого порядку?
5. Назвіть властивості теорій першого порядку.
6. Сформулюйте теорему дедукції для числення висловлювань.
7. Що таке логіки вищих порядків? На чому ґрунтуються класифікація логік?
8. Які є стандартні форми формул логіки предикатів першого порядку?

Задачі і вправи

1. Побудуйте доведення формули $(A \rightarrow B) = (\neg B \rightarrow \neg A)$, не користуючись теоремою дедукції.
2. Знайдіть для формули $\neg(A \rightarrow (C \Leftrightarrow B)) \rightarrow D$ еквівалентну їй формулу, яка містить лише логічні зв'язки: \vee і \neg , $\&$ і \neg .
3. Впевнітесь в тому, що аксіоми логіки висловлювань і аксіоми логіки предикатів — totожно істинні формули.
4. Знайдіть формулу логіки висловлювань, яка еквівалентна функції

x	y	$f(x, y)$
1	1	0
1	0	1
0	1	1
0	0	0.

5. Покажіть, що відношення R на множині всіх формул логіки висловлювань, яке визначається у вигляді $ARB \Leftrightarrow h(A) = h(B)$ для інтерпретації h , є відношенням еквівалентності.

6. Побудуйте інтерпретацію логіки висловлювань в алгебрі множин.

7. Чи буде логічним наслідком формула $\neg C \vee \neg D$ множини формул

$$(C \rightarrow G) \& (D \rightarrow S), S \& G \rightarrow E, \neg E ?$$

8. Запишіть у вигляді формул числення висловлювань наведені нижче висловлювання і знайдіть інтерпретації, при яких вони істинні.

1°. Я піду додому (H) або залишусь тут і послухаю музику (S). Я не піду додому. Значить, я залишусь тут і послухаю музику.

2°. Якщо Джон ляже спати сьогодні пізно (S), він буде вранці не в формі (D). Якщо він ляже спати не пізно, то йому буде здаватися, що не варто жити (L). Значить, або Джон буде завтра не в формі, або йому здаватиметься, що не варто жити.

3°. Заробітна плата зросте (W), якщо буде інфляція (J). Якщо буде інфляція, то збільшиться вартість життя (C). Заробітна плата зросте. Значить, збільшиться вартість життя.

4°. Якщо 2 — просте число (P), то це найменше просте число (L). Якщо 2 — найменше просте число, то 1 не є простим числом (N). Число 1 не є простим числом. Значить, 2 — просте число.

5°. Джон або перевтомився (E), або хворий (S). Якщо він перевтомився, то він дратується (C). Він не дратується. Значить, він хворий.

6°. Якщо я поїду автобусом (B), а автобус запізниться (L), то я пропущу важливе побачення (M). Якщо я пропущу важливе побачення і почну засмучуватись (D), то мені не слід їхати додому (H). Якщо я не отримаю цієї роботи (I), то я почну засмучуватись і мені треба поїхати додому. Значить, якщо я поїду автобусом і автобус запізниться, то я отримаю цю роботу.

7°. Якщо 6 — складене число (S), то 12 — складене число (W). Якщо 12 — складене число, то існує просте число, більше, ніж 12 (P). Якщо існує просте число, більше, ніж 12, то існує складене число, більше 12 (C). Якщо 6 ділиться на 2 (D), то 6 — складене число. Число 12 складене. Отже, 6 — складене число.

8°. Якщо завтра буде холодно (C), я одягну тепле пальто (H), якщо рукав буде полагоджений (S). Завтра буде холодно, а рукав не буде полагоджений, значить, я не одягну тепле пальто.

9. Нехай $P(x)$ означає “ x — просте число”, $E(x)$ — “ x — парне число”, $O(x)$ — “ x — непарне число”, $D(x, y)$ — “ y ділиться на x ”. Перекладіть на українську мову такі формули логіки предикатів першого порядку:

- а) $P(7)$;
- б) $E(2) \& P(2)$;

- в) $\forall x (E(x) \ \& \ D(x, 6));$
- г) $\forall x (\neg E(x) \rightarrow \neg D(2, x));$
- д) $\forall x (E(x) \ \& \ \forall y (D(x, y) \rightarrow E(y)));$
- е) $\forall x (P(x) \rightarrow (\exists y) (E(y) \ \& \ D(x, y)));$
- ж) $\forall x (O(x) \rightarrow (\forall y) (P(y) \rightarrow \neg D(x, y)));$
- з) $(\exists x) (E(x) \ \& \ P(x)) \ \& \ \neg(\exists x) ((E(x) \ \& \ P(x)) \ \&$
 $\quad \& \ ((\exists y) (x \neq y \ \& \ E(x) \ \& \ P(y))).$

10. Нижче наведено п'ять речень українською мовою, за якими іде стільки ж речень символічної мови предикатів першого порядку. Знайдіть відповідні пари речень, так щоб кожний член пари був перекладом члена, який йому відповідає.

- 1. Всі судді ($J(x)$) — юристи ($L(x)$).
 - 2. Деякі юристи — шахраї ($S(x)$).
 - 3. Не всі юристи — судді.
 - 4. Жоден суддя не є шахраєм.
 - 5. Деякі юристи — політики ($P(x)$).
- 1'. $\exists x (L(x) \ \& \ S(x)).$
 - 2'. $\exists x (L(x) \ \& \ P(x)).$
 - 3'. $\neg(\forall x) (L(x) \rightarrow J(x)).$
 - 4'. $\forall x (J(x) \rightarrow L(x)).$
 - 5'. $\forall x (L(x) \rightarrow \neg S(x)).$

11. Вкажіть вільні і зв'язані входження змінних у таких формулах:

- а) $\forall z (\forall x A(x, y) \rightarrow B(z, x));$
- б) $\forall y A(z, y) \rightarrow \forall z A(z, y);$
- в) $(\forall y \exists y (A(x, y, f(x, y))) \vee \neg \forall x B(y, f(x)).$

12. Перекладіть на мову формул такі речення:

- а) не всі птахи можуть літати;
 - б) або кожний любить кого-небудь, і ні один не любить всіх, або хтось любить всіх, і хтось не любить нікого;
 - в) якщо хтось може це зробити, то і я теж можу це зробити;
 - г) не всі люди щирі і не всі щирі люди багаті.
13. Чи вільний терм $f(x, y)$ для x у формулах $A(x, y) \rightarrow \forall x B(x)$, $(\forall y A(y, a)) \vee \exists y A(x, y)$?

Розділ 4

ЕЛЕМЕНТИ ТЕОРИЇ ГРАФІВ

Виникнення теорії графів пов'язують з іменем Ейлера, який у 1736 р. не тільки розв'язав популярну на той час головоломку про кенігсберзькі мости, а й знайшов критерій існування в графі спеціального маршруту (ейлерового циклу). Довгий час цей результат залишався єдиним результатом теорії графів, і лише в середині XIX ст., переважно зусиллями Кірхгофа та Келлі, були одержані нові результати в теорії графів. Приблизно в цей же час виникла знаменита проблема чотирьох фарб.

Хоча теорія графів виникла більше двох століть тому, але її інтенсивний розвиток припадає лише на останні 50—60 років. Цей розвиток завдячує широкому застосуванню графів у теорії автоматів, теорії проектування, економіці, хімії, біології і т.д. Оскільки 50—60 років для глибокої теорії — це відносно молодий вік, то термінологія в теорії графів ще досі змінюється. Виникають нові поняття і методи. У зв'язку з цим понятійний матеріал розділу може мати деякі розбіжності з іншими літературними джерелами з теорії графів, хоча в ньому зібрани більш менш усталені поняття, означення і методи теорії графів.

§ 4.1. ОЗНАЧЕННЯ ГРАФІВ, РІЗНОВИДИ ГРАФІВ

1. ОЗНАЧЕННЯ НЕОРІЄНТОВАНОГО ГРАФА

Нехай V — деяка непуста множина. Позначимо $V^{(2)}$ — множину всіх неупорядкованих різних двохелементних підмножин множини V , а $MV^{(2)}$ — мультимножину множини $V^{(2)}$, тобто в $MV^{(2)}$ можуть входити однакові пари елементів із V , причому цих пар може бути скільки завгодно. Як і раніше, декартовий квадрат множини V позначатимемо V^2 .

Неорієнтованим мультиграфом G називається пара (V, E) , де $E \subseteq MV^{(2)}$. Елементи множини V називаються *вершинами*, а еле-

менти множини E — *ребрами*. Ребра позначаються парами (u, v) , де u, v — вершини з V .

Мультиграф $G = (V, E)$ називається *неорієнтованим графом*, якщо $E \subseteq V^{(2)}$. Підкреслимо, що всякий граф є мультиграфом, але не всякий мультиграф буде графом. Якщо $G = (V, E)$ — мультиграф, то E може мати кілька ребер, що з'єднують одні і ті ж вершини u і v . Такі ребра називаються *кратними ребрами*. Отже, граф є мультиграфом, в якого кратність кожного ребра дорівнює одиниці.

Мультиграф називається *скінченим*, якщо множини V і E скінченні. Для скінченності графа, очевидно, достатньо лише скінченності множини його вершин V , оскільки скінченність V дає скінченність $V^{(2)}$, тобто граф називається *скінченим*, якщо скінчена множина його вершин. Скінчений граф з n вершинами називається *графом n -го порядку*.

Інколи розглядають графи, які мають ребра (u, u) . Таке ребро називається *петлею*, а мультиграф, який має петлі, — *псевдографом*.

Надалі, коли не сказано протилежне, будемо розглядати тільки скінченні графи.

Говорять, що дві вершини u і v графа $G = (V, E)$ *суміжні*, якщо $(u, v) \in E$, і *несуміжні* — в протилежному випадку. Якщо $(u, v) \in E$, то вершини u і v називаються *кінцями ребра* (u, v) . У цьому випадку ще говорять, що ребро (u, v) з'єднує вершини u і v . Множину вершин графа, суміжних з деякою вершиною u , будемо позначати $\text{См}(u)$.

З цих означенень випливає також, що різниця між графом і мультиграфом полягає в тому, що дві вершини в графі можуть бути з'єднані не більше, ніж одним ребром, а в мультиграфі дві вершини можуть з'єднуватись більше, ніж одним ребром.

Два ребра називаються *суміжними*, якщо вони мають спільний кінець. Зауважимо, що відношення суміжності як для вершин, так і для ребер є симетричним відношенням.

Вершина u і ребро e називаються *інцидентними*, якщо u є кінцем ребра e , і *неінцидентними* — в протилежному випадку.

Степенем $n(u)$ вершини u графа G називається число інцидентних їй ребер. Вершина степені 0 називається *ізольованою*, а вершина степені 1 — *висячою*, або *кінцевою*. Ребро, інцидентне кінцевій вершині, також називається кінцевим. Зручну характеристику графів дає таке просте, майже очевидне, твердження.

Лема про рукостискання. *Сума степенів всіх вершин графа є парним числом.*

Дійсно, кожне ребро вносить у суму всіх вершин графа число 2, тобто

$$\sum_{v \in V} n(v) = 2|E|.$$

Якщо інтерпретувати кожне ребро як рукостискання двох людей, то отримаємо, що при будь-якому числі рукостискань загальна кількість потиснутих рук буде парною.

Наслідок 4.1.1. У будь-якому графі число вершин непарного степеня парне.

Дійсно, якби це було не так, то сума степенів всіх вершин графа не могла б бути парним числом, що суперечить лемі про рукостискання.

Графи зручно зображувати на площині або в просторі у вигляді діаграм, які складаються з точок і відрізків, що з'єднують деякі з цих точок. При цьому точки ототожнюються з вершинами графа, а відрізки — з його ребрами (див. рис. 4.1.1, а, б, в).

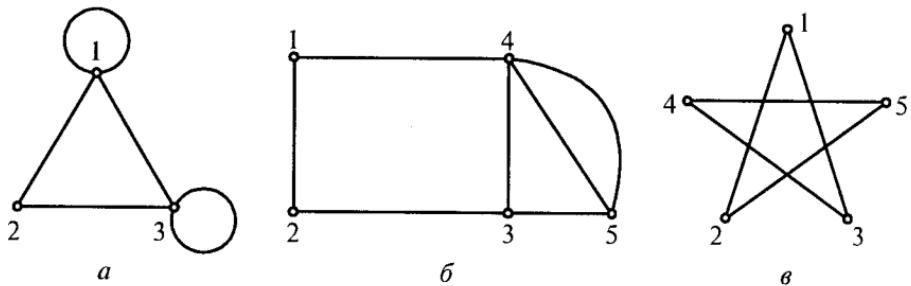


Рис. 4.1.1. Зображення графів: а — псевдограф; б — мультиграф; в — граф

2. РІЗНОВИДИ ГРАФІВ

Розглянемо деякі різновиди графів, які часто зустрічаються.

2.1. Повні графи. Граф, будь-які дві вершини якого суміжні, називається **повним графом**. Отже, якщо $G = (V, E)$ — повний граф, то $E = V^{(2)}$. Повний граф з n вершинами позначаємо K_n . Графи K_4 і K_5 зображені на рис. 4.1.2 і 4.1.3 відповідно.

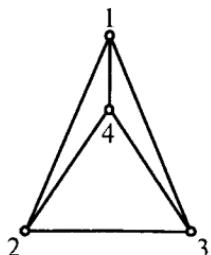


Рис. 4.1.2. Граф K_4

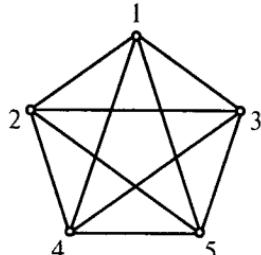


Рис. 4.1.3. Граф K_5

2.2. Регулярні графи. Більш загальними, ніж повні, є регулярні графи. Граф називається *регулярним* або *однорідним*, якщо всі його вершини мають один і той же степінь. Якщо степінь кожної вершини дорівнює k , то граф називається регулярним графом ступеня k . Отже, повний граф n -го порядку є регулярним графом ступеня $n - 1$. Регулярні графи ступеня 3 називають також *кубічними*, або *тривалентними графами* (ці графи викликають особливий інтерес у зв'язку із задачею розфарбування графів, яку будемо розглядати пізніше). Відомим прикладом кубічного графа є граф Петерсена, який показано на рис. 4.1.4.

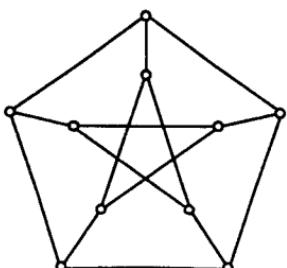


Рис. 4.1.4. Граф Петерсена

2.3. Повністю незв'язні графи. Граф, множина ребер якого пуста, називається *повністю незв'язним* або *пустим*. Будемо позначати повністю незв'язні графи з n вершинами N_n . На рис. 4.1.5 показаний граф N_4 .

Зауважимо, що кожний пустий граф є регулярним графом ступеня 0.



Рис. 4.1.5. Пустий граф

2.4. Платонові графи. *Платоновими графами* називаються графи, утворені вершинами і ребрами п'яти правильних многогранників — платонових тіл: тетраедра, куба, октаедра, додекаедра та ікосаедра. Граф K_4 , зображений на рис. 4.1.2, відповідає тетраедру, а графи, що відповідають кубу і октаедру, показані на рис. 4.1.6 і 4.1.7 відповідно.

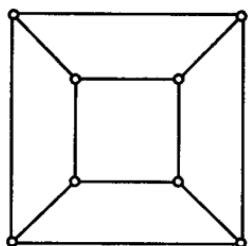


Рис. 4.1.6. Граф, що відповідає кубу

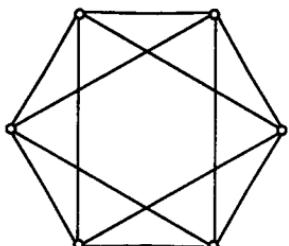


Рис. 4.1.7. Граф, що відповідає октаедру

Решту графів, які відповідають додекаедру та ікосаедру, пропонується намалювати читачеві (див. вправу 1 в кінці параграфа).

2.5. Двочастинні графи. Граф називається **двочастинним**, якщо існує таке розбиття множини його вершин на два класи, при якому кінці кожного ребра лежать у різних класах. Означення двочастинного графа можна подати іншим чином — в термінах розфарбування його вершин двома кольорами, наприклад червоним і синім. При цьому граф називається **двочастинним**, якщо кожну його вершину можна пофарбувати синім або червоним кольором, так щоб кожне його ребро мало один кінець червоний, а другий — синій.

Якщо в двочастинному графі будь-які дві вершини з різних класів суміжні, то такий граф називається **повним двочастинним графом**. Повний двочастинний граф, в якого один клас має m

вершин, а другий — n вершин, позначають $K_{m,n}$.

Повний двочастинний граф $K_{1,n}$ називається **зірковим графом**. На рис. 4.1.8 зображеній граф $K_{1,5}$.

Аналогічно можна ввести k -частинні графи. Граф називається **k -частинним графом**, якщо існує таке розбиття множини його вершин на k класів, при якому будь-яке ребро графа з'єднує дві вершини з різних класів.

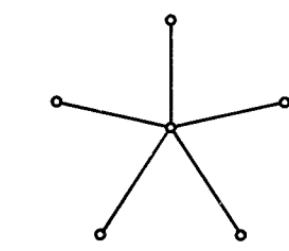


Рис. 4.1.8. Граф $K_{1,5}$

2.6. Орієнтовані графи (орграфи). Граф $G = (V, E)$ називається **орієнтованим графом (орграфом)**, якщо $E \subseteq V^2$, тобто вершини всіх його ребер упорядковані. Якщо $(u, v) \in E$, то вершину u називають **початковою** вершиною, а v — **кінцевою** вершиною ребра. Орграфи зображуються так як, і графи, з тією лише різницею, що їх ребра позначаються стрілками, які ведуть з початкової вершини ребра в кінцеву. Іншими словами, якщо (u, v) — ребро

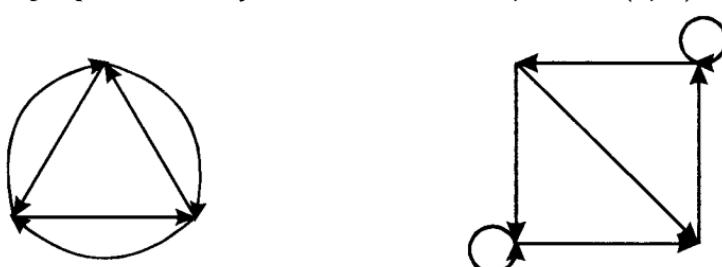


Рис. 4.1.9. Орграфи

ографа, то з вершини u у вершину v веде стрілка. Приклади орграфів показані на рис. 4.1.9.

Можна дати означення і орієнтованого мультиграфа. Граф $G = (V, E)$ називається *орієнтованим мультиграфом*, якщо $E \subseteq MV^2$ і кожне його ребро упорядковане, тобто вказано, яка вершина перша, а яка друга. Отже, в орграфах ребра (u, v) і (v, u) різні.

3. ІЗОМОРФІЗМ ГРАФІВ. ПІДГРАФИ

Нехай $G = (V, E)$, $H = (V_1, E_1)$ — графи і $h: V \rightarrow V_1$ — взаємно однозначна відповідність (тобто $|V| = |V_1|$). Відображення h називається *ізоморфізмом графів* G і H , якщо для будь-яких вершин u і v графа G їх образи $h(u)$ і $h(v)$ суміжні в графі H тоді і тільки тоді, коли u і v суміжні в G . Якщо таке відображення h існує, то графи G і H називаються *ізоморфними*.

Очевидно, що відношення ізоморфізму графів є відношенням еквівалентності. Ізоморфні графи, як правило, не розрізняються.

Граф $H = (V', E')$ називається *підграфом* графа $G = (V, E)$, якщо $V' \subseteq V$ і $E' \subseteq E$. Якщо H — підграф графа G , то говорять, що H знаходиться в графі G . Підграф H графа G називається *остовним підграфом*, коли $V' = V$.

Приклади 4.1.1

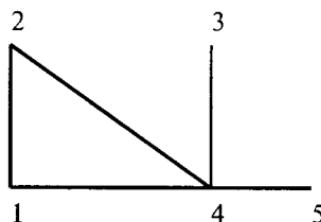
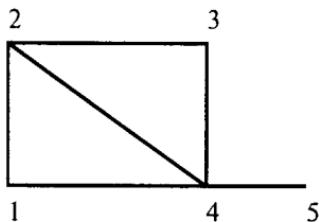


Рис. 4.1.10. Граф і його оставний підграф

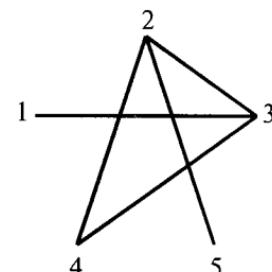
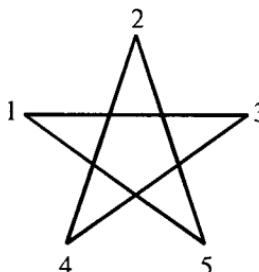
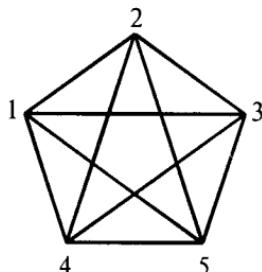


Рис. 4.1.11. Граф і його підграфи. ▲

Контрольні питання

1. Що називається графом, мультиграфом і псевдографом?
2. Яка різниця між графом і псевдографом?
3. Який граф називається регулярним, регулярним степеня k , повним?
4. Чи являється:
 - а) повний граф регулярним графом?
 - б) регулярний граф повним?
5. Чи буде скінченним граф, який має скінченне число вершин, ребер?
6. Які бувають різновиди графів?

Задачі і вправи

1. Побудуйте платонові графи для ікосаедра і додекаедра.
2. Чи будуть платонові графи регулярними?

§ 4.2. ОПЕРАЦІЇ НАД ГРАФАМИ

На практиці часто зустрічаються графи, які будується з деякого початкового графа за допомогою вилучення однієї з його вершин або одного з його ребер. Існують і інші можливі перетворення графів, які розглядаються як операції над графами. Розглянемо основні операції над графами і деякі їх властивості.

1. ОПЕРАЦІЯ ВИЛУЧЕННЯ РЕБРА

Нехай $G = (V, E)$ — граф і $e \in E$ — деяке його ребро. Говорять, що граф $G_1 = G - e$ одержано з графа G внаслідок операції вилучення ребра e , якщо $G_1 = (V, E \setminus \{e\})$. Отже, кінці ребра e не вилучаються з множини V .

Неважко показати, що для довільних ребер e і e_1 графа G виконується така тотожність:

$$(G - e) - e_1 = (G - e_1) - e.$$

Дійсно, оскільки виконується тотожність $(A \setminus B) \setminus C = A \setminus (B \cup C)$ (див. вправу 10 із § 1.1), то маємо

$$\begin{aligned} G_1 &= G - e = (V, E \setminus \{e\}), \\ (G - e) - e_1 &= G_1 - e_1 = (V, (E \setminus \{e\}) \setminus \{e_1\}) = (V, E \setminus (\{e\} \cup \{e_1\})) = \\ &= (V, E \setminus (\{e_1\} \cup \{e\})) = (V, (E \setminus \{e_1\}) \setminus \{e\}) = (G - e_1) - e. \end{aligned}$$

Отже, якщо виконується підряд кілька операцій вилучення ребра, то результат не залежить від порядку вилучення ребер з графа.

Приклад 4.2.1

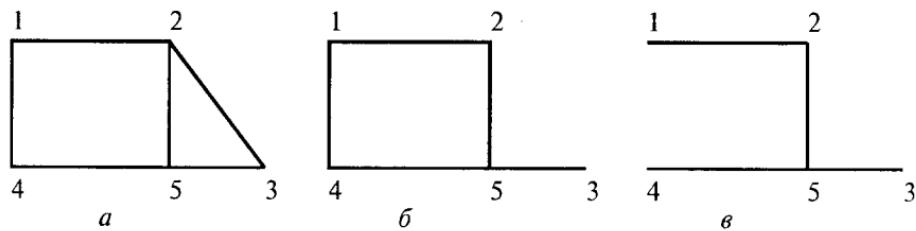


Рис. 4.2.1. Вилучення ребра:

a — граф G ; b — граф $G \setminus \{(2, 3)\}$; v — граф $(G \setminus \{(2, 3)\}) \setminus \{(1, 4)\}$. ▲

2. ОПЕРАЦІЯ ВИЛУЧЕННЯ ВЕРШИНІ

Нехай $G = (V, E)$ і $v \in V$. Говорять, що граф $G_v = G - v$ одержаний з графа G внаслідок операції вилучення вершини v , якщо вершина v вилучена з V , а з E вилучені всі ребра, інцидентні з вершиною v .

Неважко переконатися, що операція вилучення вершини не залежить від порядку, в якому вилучаються вершини з графа (див. вправу 7 в кінці розділу).

Приклад 4.2.2

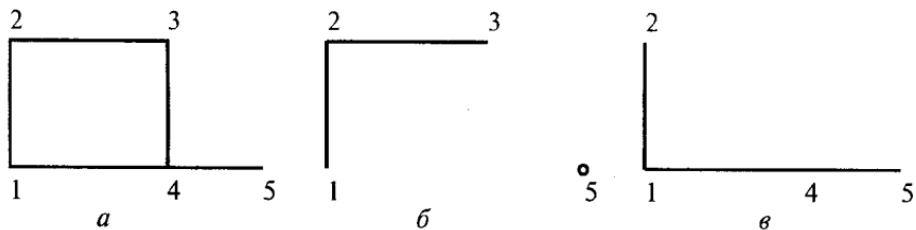


Рис. 4.2.2. Вилучення вершини:

a — граф G ; b — граф $G - \{4\}$; v — граф $G - \{3\}$. ▲

Операції вилучення ребра, вершини і переходу до підграфа — це операції, за допомогою яких можна з початкового графа одержувати графи з меншим числом вершин і ребер. Є й інші опера-

ції, які дають можливість будувати з початкових графів нові графи з більшим числом вершин і ребер.

3. ОПЕРАЦІЯ ВВЕДЕННЯ РЕБРА

Якщо $u, v \in V$ і $(u, v) \notin E$ в графі $G = (V, E)$, то граф $G + e = (V, E \cup \{e\})$, де $e = (u, v)$.

Внаслідок комутативності операції об'єднання множин можна стверджувати, що послідовність операцій введення ребер у граф G не залежить від порядку, в якому ці ребра вводяться в граф G . Іншими словами, справедлива тотожність

$$\forall e, e_1 \in E \quad ((G + e) + e_1 = (G + e_1) + e).$$

4. ОПЕРАЦІЯ ВВЕДЕННЯ ВЕРШИНИ В РЕБРО

Нехай (u, v) — деяке ребро графа G . *Введенням вершини w в ребро (u, v)* називається операція, внаслідок якої одержуємо два ребра (u, w) і (w, v) , а ребро (u, v) при цьому вилучається з графа G .

5. ОПЕРАЦІЯ ОБ'ЄДНАННЯ ГРАФІВ

Граф F називається об'єднанням графів $G = (V, E)$ і $H = (V_1, E_1)$, якщо $F = (V \cup V_1, E \cup E_1)$. Граф F позначається $G \cup H$. Об'єднання графів $F = G \cup H$ називається *диз'юнктивним*, якщо $V \cap V_1 = \emptyset$.

Безпосередньо з означення операції об'єднання графів випливає, що $(\forall G, H) (G \cup H = H \cup G)$.

Операція диз'юнктивного об'єднання графів дає можливість ввести до розгляду ще один важливий тип графів.

Граф називається *зв'язним*, якщо його не можна подати у вигляді диз'юнктивного об'єднання двох підграфів, і *незв'язним* — у протилежному випадку.

Отже, всякий незв'язний граф можна зобразити у вигляді диз'юнктивного об'єднання скінченного числа зв'язних підграфів. Кожний із таких зв'язних підграфів називається *компонентом зв'язності*.

Зв'язний регулярний граф степеня 2 називається *циклічним графом*. Циклічний граф з n вершинами позначається C_n . Циклічний граф C_6 наведено на рис. 4.2.3.

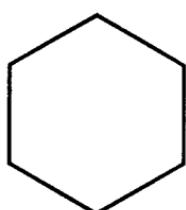


Рис. 4.2.3.
Циклічний граф

6. ДОБУТОК ГРАФІВ

Добутком графів $G = (V, E)$ і $H = (V_1, E_1)$ називається граф $F = G \times H$, у якого $V = V \times V_1$, а E визначається таким чином: вершини (u, u_1) і (v, v_1) суміжні в F тоді і тільки тоді, коли $u = v$, а u_1 і v_1 суміжні в H або $u_1 = v_1$, а u і v суміжні в G .

Приклад 4.2.3

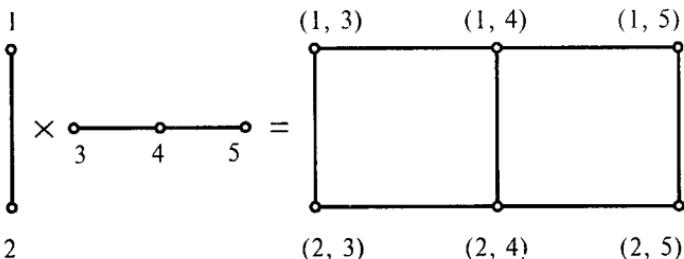


Рис. 4.2.4. Добуток графів. ▲

7. ОТОТОЖНЕННЯ (ЗЛІТТЯ) ВЕРШИН

Якщо $G = (V, E)$ — граф, u, v — дві його вершини і $\text{См}(u) = \{u_1, \dots, u_k\}$, $\text{См}(v) = \{v_1, \dots, v_l\}$, то граф $H = G - u - v$, одержаний приєднанням нової вершини u' до множини вершин H і множини ребер $(u', u_i), (u', v_j)$ ($i = 1, 2, \dots, k, j = 1, 2, \dots, l$) до множини ребер H , називається графом, одержаним із G ототожненням вершин u і v .

8. ОПЕРАЦІЯ СТЯГУВАННЯ РЕБРА

Операція стягування ребра (u, v) в графі $G = (V, E)$ означає ототожнення вершин u і v в графі G . Операція стягування ребра дозволяє ввести таке поняття. Граф G називається графом, який **стягується до графа** H , якщо H можна одержати з G за допомогою деякої послідовності операцій стягування ребра. Легко помітити, наприклад, що граф Петерсена стягується до графа K_5 і, значить, до всякого графа K_n , де $n < 5$. Очевидно також, що всякий непустий зв'язний граф, відмінний від K_1 , стягується до K_2 . Але вже не всякий зв'язний граф стягується до K_3 . Наприклад, простий ланцюг P_n не стягується до K_3 . Логічно ввести параметр $\chi(G)$ — максимум порядків повних графів, до яких стягується граф G . Параметр $\chi(G)$ називається **числом Хадвігера** графа G .

Приклад 4.2.4

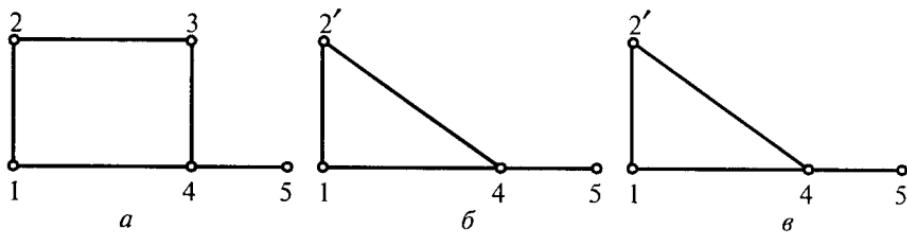


Рис. 4.2.5. Стягування ребра:
 a — граф G ; b — ототожнення вершин 2 і 3 в G ;
 c — стягування ребра $(2, 3)$ в G .

9. ОПЕРАЦІЯ РОЗДВОЄННЯ (РОЗЩЕПЛЕННЯ) ВЕРШИНИ

Нехай v — деяка з вершин графа G . Розіб'ємо множину суміжних з нею вершин довільним чином на дві частини — M і P , а потім виконаємо таке перетворення графа G : вилучимо вершину v разом з інцидентними їй ребрами і введемо дві нові вершини u і w разом з ребром, яке з'єднує ці вершини, вершину u з'єднаємо ребром з кожною вершиною множини M , а вершину w — з кожною вершиною з множини P . Одержаній граф позначимо \tilde{G} і будемо вважати, що він одержаний з графа G внаслідок роздвоєння (розщеплення) вершини v (рис. 4.2.6).

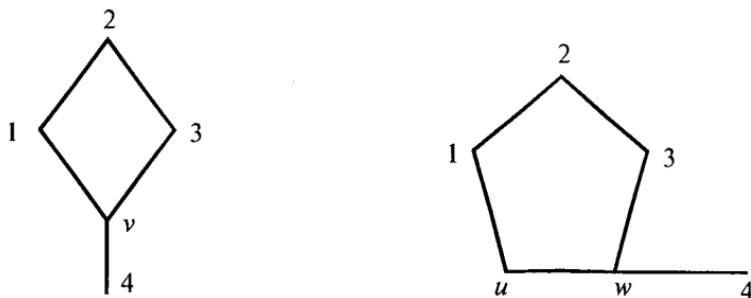


Рис. 4.2.6. Роздвоєння вершини v

10. ОПЕРАЦІЯ З'ЄДНАННЯ ГРАФІВ

Нехай $G = (V, E)$ і $G_1 = (V_1, E_1)$ — два графи, у яких множини вершин V і V_1 не перетинаються, тобто $V \cap V_1 = \emptyset$. **Операція з'єднання** графів G і G_1 полягає в тому, що множини V і V_1 об'єднуються, а потім з'єднуються ребрами кожна вершина графа G з кожною вершиною графа G_1 . Іншими словами, якщо E'' означає

множину ребер, яка одержана об'єднанням множин E і E_1 разом з утвореними новими ребрами, то $G = G + G_1 = (V \cup V_1, E'')$.

Очевидно, що операція з'єднання графів може бути виражена у вигляді добутку (суперпозиції) операції об'єднання графів G і G_1 та послідовності операцій введення ребра.

З'єднання графів N_1 і C_{n-1} ($n > 3$) називається **колесом** з n вершинами і позначається W_n . На рис. 4.2.7 зображені графи C_6 і W_7 .

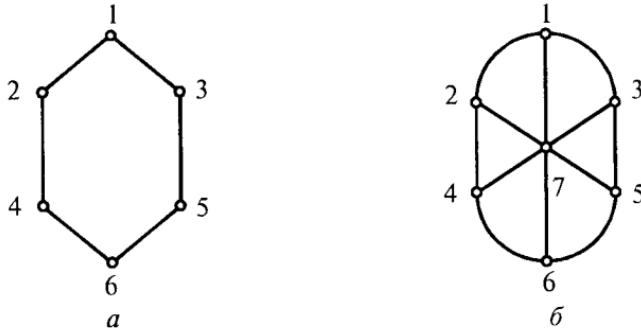


Рис. 4.2.7. З'єднання графів:
а — граф C_6 ; б — граф W_7

11. ОПЕРАЦІЯ ДОПОВНЕННЯ ГРАФА

Нехай $G = (V, E)$ — граф. **Доповненням** G^* графа G називається граф з множиною вершин V , в якому дві вершини суміжні тоді і тільки тоді, коли вони не суміжні в графі G . Звідси випливає, що коли граф G має n вершин, то граф G^* можна побудувати, вилучивши з графа K_n всі ребра, які належать G (граф G вважається підграфом графа K_n). Очевидно також, що доповнення повного графа є пустим графом і, навпаки, доповнення пустого графа є повним графом. Неважко довести, що доповнення регулярного графа є регулярним графом.

Приклад 4.2.5



Рис. 4.2.8. Доповнення графа:
а — граф G ; б — доповнення графа G .

§ 4.3. ВЛАСТИВОСТІ ГРАФІВ

Розглянемо детальніше деякі властивості графів і пов'язаних з ними понять.

1. МАРШРУТИ, ЦИКЛИ, ЗВ'ЯЗНІСТЬ

Маршрутом у заданому графі $G = (V, E)$ називається скінчена послідовність його ребер, яка має вигляд

$$(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k).$$

Число k ребер маршруту називається довжиною цього маршруту. Часто можна зустріти і таке означення маршруту в графі: послідовність вершин v_0, v_1, \dots, v_k графа $G = (V, E)$ називається **маршрутом**, який з'єднує вершини v_0 і v_k , якщо $(v_i, v_{i+1}) \in E$, $i = 0, 1, \dots, k-1$. В обох випадках вершини v_0, v_1, \dots, v_k називаються **вершинами маршруту**.

Очевидно, що відношення **маршрут**, який з'єднує вершини..., є симетричним і транзитивним.

Маршрут називається **ланцюгом**, якщо всі його ребра різні, і **простим ланцюгом**, якщо всі його вершини, крім, можливо, першої і останньої, різні. Маршрут називається **циклічним**, якщо перша і остання його вершини збігаються.

Циклом називається циклічний ланцюг, а **простим циклом** — простий циклічний ланцюг.

Граф називається **ациклічним графом** або **лісом**, якщо в ньому відсутні цикли. Безпосередньо з означення маршруту і циклу випливають такі твердження.

Твердження 4.3.1. *Будь-який маршрут, який з'єднує які-небудь дві вершини графа, має простий ланцюг, що з'єднує ці вершини.*

Твердження 4.3.2. *Будь-який цикл в графі має простий цикл.*

Тепер можна дати більш зручне означення зв'язного графа.

Граф називається **зв'язним**, якщо довільні дві його вершини зв'язані маршрутом (а згідно з твердженням 4.3.1 можна сказати — і простим ланцюгом). Зв'язний підграф H графа G називається **максимальним**, якщо H не міститься в жодному зв'язному підграфі графа G . Максимальний зв'язний підграф графа називається **компонентом зв'язності**. Еквівалентність двох означень зв'язного графа встановлює така теорема.

Теорема 4.3.1. *Граф зв'язний тоді і тільки тоді, коли його не можна представити у вигляді диз'юнктивного об'єднання двох графів.*

Д о в е д е н н я. Нехай граф зв'язний, тобто всякі дві його вершини *u* і *v* зв'язані маршрутом. Припустимо, що граф G являє

собою діз'юнктивне об'єднання двох підграфів, і вершини u і v належать різним підграфам. Тоді всякий простий ланцюг, який з'єднує вершини u і v , повинен мати ребро, інцидентне деяким двом вершинам із різних підграфів. Але такого ребра не існує за нашим припущенням.

Припустимо тепер, що граф G не можна подати у вигляді об'єднання двох підграфів, і не існує жодного простого ланцюга, який з'єднував би задану пару вершин u і v . Тоді вершини u і v належать різним компонентам зв'язності. А це означає, що граф G можна подати у вигляді об'єднання двох підграфів, — компонента зв'язності, якому належить вершина u , і об'єднання решти компонентів.

Одержані суперечності остаточно доводять теорему.

2. ВЛАСТИВОСТІ РЕГУЛЯРНИХ ГРАФІВ

Нехай G — регулярний граф степеня k . Степінь регулярного графа позначають $\deg(G)$. Очевидним наслідком означення регулярного графа є таке твердження.

Твердження 4.3.3. *Повний граф є регулярним графом.*

Користуючись лемою про рукостискання, неважко встановити справедливість такого простого твердження для регулярних графів.

Твердження 4.3.4. *Не існує регулярного графа $G = (V, E)$ з n вершинами степеня k , у якого k і n непарні.*

Доведення. Дійсно, якщо n і k — непарні числа, то їх добуток $n \cdot k$ теж непарне число, але $n \cdot k = 2 \cdot |E|$ за лемою про рукостискання. Отримана суперечність доводить наше твердження.

Теорема 4.3.2. *Нехай $n, d \in N$ — натуральні числа, одне з яких парне і для яких виконується нерівність $0 \leq d \leq n - 1$. Тоді існує регулярний граф порядку n і степеня d .*

Доведення. Для $d = 0$ твердження теореми очевидне. Крім того, якщо G — регулярний граф порядку n і степеня d , то його доповнення — граф G^* — теж регулярний граф і $\deg(G^*) = n - 1 - d$. У зв'язку з цим достатньо розглянути випадок, коли $0 < d \leq (n - 1)/2$.

Нехай Z_n — адитивна група лижків цілих чисел за модулем n , $0 \notin A$ і для $x \in A$ x -клас також належить множині A . Визначимо граф G порядку n з множиною вершин Z_n такою умовою: вершини x і y суміжні, якщо $x - y \in A$. Очевидно, що граф G регулярний і степінь його дорівнює $|A|$. Залишається лише довести, що для всякого числа d , яке задовільняє умови теореми, існує від-

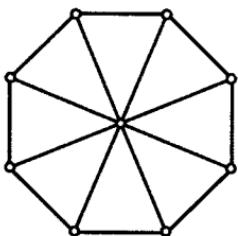


Рис. 4.3.1.
Регулярний граф

Д о в е д е н н я. Оскільки множині V належить лише один кінець кожного з ребер графа G , то число m його ребер дорівнює $|V| \cdot \deg(G)$. Аналогічно $m = |V_1| \cdot \deg(G)$. Отже, $|V| \cdot \deg(G) = |V_1| \cdot \deg(G)$. Оскільки G не пустий, то $\deg(G) \neq 0$ і, значить, $|V| = |V_1|$.

3. ВЛАСТИВОСТІ ДВОЧАСТИННИХ ГРАФІВ

Існує простий критерій двочастинності графа, який виражається в термінах довжини циклів.

Теорема 4.3.4 (Кьюніг). *Граф $G = (V, E)$ є двочастинним тоді і тільки тоді, коли він не має циклів непарної довжини.*

Д о в е д е н н я. Нехай G — двочастинний граф і C — один із його циклів довжини k . Пройдемо всі ребра цього циклу в тій послідовності, в якій вони зустрічаються в ньому, починаючи з деякої вершини v . Пройшовши k ребер, повертаємося знову у вершину v . Оскільки кінці кожного ребра лежать у різних частинах, то k — парне число. Нехай тепер G — зв'язний граф n -го порядку ($n > 1$), не має циклів непарної довжини і $v \in V$. Побудуємо розбиття $V = V_1 \cup V_2$ таким чином: довільну вершину v графа G віднесемо до класу V_1 , якщо відстань $d(v, u)$ — парне число, і віднесемо її до класу V_2 , якщо ця відстань — непарне число. Покажемо тепер, що підграфи $G_1 = (V_1, E_1)$ і $G_2 = (V_2, E_2)$, які породжуються множинами вершин V_1 і V_2 відповідно, є пустими (вершини з V_1 і з V_2 не з'єднані між собою жодним ребром). Припустимо, що це не так, тобто що існують суміжні вершини u і w , які належать одному і тому ж класу. Тоді жодна з них не збігається з вершиною v . Нехай P — найкоротший ланцюг, який з'єднує вершини u і v , а Q — найкоротший ланцюг, який з'єднує вершини u і w , w — найкоротший (w, v) шлях, і нехай v_1 — остання вершина, якщо відлік вести від вершини v . Вона спільна для ланцюгів P і Q і належить ланцюгу P (рис. 4.3.2). Позначимо X_u і Y_u відповідно підланцюги, які з'єднують

повідна d -елементна множина A . При $d = 2k$ можна взяти $A = \{\pm 1, \pm 2, \dots, \pm k\}$, а при $d = 2k + 1$ число n парне і можна взяти $A = \{\pm 1, \pm 2, \dots, \pm k, n/2\}$ (див. рис. 4.3.1, де $n = 8$, $d = 3$).

Теорема доведена.

Теорема 4.3.3. *Нехай $G = (V \cup V_1, E)$ — непустий регулярний двочастинний граф. Тоді $|V| = |V_1|$, де V, V_1 — класи розбиття множини вершин графа G .*

вершини v і v_1 та v_i і u ланцюга P , а X_u і Y_w — відповідно підланцюги ланцюга Q , які з'єднують вершини v і v_1 та v_1 і w .

Очевидно, що довжини ланцюгів X_v і X_w рівні і, значить, довжини ланцюгів Y_u і Y_w або парні, або непарні. Але тоді об'єднання ланцюгів Y_u і Y_w і ребра (v, w) є циклом непарної довжини. Одержано суперечність доводить теорему.

Наслідок 4.3.1. Граф двочастинний тоді і тільки тоді, коли він не має простих циклів непарної довжини.

З доведення теореми 4.3.4 випливає такий простий спосіб розпізнавання двочастинності графа. Будемо приписувати номери 0 і 1 вершинам графа G :

- починаючи з довільної вершини u графа G , приписуємо їй номер 0;
- кожній вершині з множини $C_m(u)$ приписуємо номер 1;
- для всіх вершин, суміжних з вершинами множини $C_m(u)$, приписуємо номер 0;
- для всіх вершин, яким приписали номер 0, знаходимо всі суміжні з ними вершини і приписуємо їм номер 1 і т.д.

Після того як всі вершини будуть перенумеровані таким чином, будуємо дві множини: V_0 і V_1 , до яких входять всі вершини з номерами відповідно 0 і 1. Якщо графи $G_1 = (V_0, E_0)$ і $G_2 = (V_1, E_1)$ пусті, то граф G двочастинний, а якщо ні (тобто або E_0 , або E_1 непусті), то G не є двочастинним.

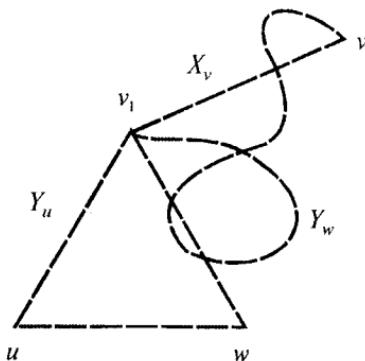


Рис. 4.3.2. Ланцюг P

4. ВЛАСТИВОСТІ ЗВ'ЯЗНИХ ГРАФІВ

Вияснимо тепер деякі властивості зв'язних графів. Насамперед встановимо такий факт.

Теорема 4.3.5. *Будь-який граф $G = (V, E)$ єдиним способом подається у вигляді диз'юнктивного об'єднання своїх компонентів зв'язності.*

Д о в е д е н н я. Визначимо на множині вершин графа G відношення R : $uRv \Leftrightarrow (u = v)$, тобто існує маршрут в G , який з'єднує вершини u і v .

Неважко перевірити, що відношення R являє собою відношення еквівалентності. Отже, множина вершин графа розбива-

ється на класи, що не перетинаються. Нехай $V = \bigcup_{i=1}^k V_i$ — розбиття. Тоді підграфи $G_i = (V_i, E_i)$, де E_i — множина ребер у графі G , що з'єднують вершини з V_i в графі G , і тільки вони є компонентами зв'язності графа G і $G = \bigcup_{i=1}^k G_i$ — диз'юнктивне об'єднання.

Теорема доведена.

Часто виникає питання про число ребер зв'язного графа.

Нехай $G = (V, E)$ — граф з n вершинами і k компонентами зв'язності. Якщо такий граф зв'язний, то природно чекати, що число ребер у ньому мінімальне, коли він ацикличний, і максимальне, коли він повний. Звідси випливає така оцінка для числа ребер зв'язного графа:

$$n - 1 \leq |E| \leq n \cdot (n - 1) / 2.$$

Насправді існує сильніший результат.

Теорема 4.3.6. *Нехай G — граф з n вершинами і k компонентами зв'язності. Тоді число m його ребер задовольняє нерівності*

$$n - k \leq m \leq (n - k) \cdot (n - k + 1) / 2.$$

Д о в е д е н н я. Нерівність $m \geq n - k$ легко довести методом математичної індукції. Дійсно, якщо G повністю незв'язний граф, то нерівність справедлива: у цьому випадку $k = n$, $m = 0$ і $0 \geq n - k = 0$. Нехай G має мінімальне число ребер, наприклад m' . Тоді вилучення одного ребра приводить до збільшення компонентів зв'язності на одиницю. Тобто одержаний граф буде мати n вершин, $k + 1$ компонент зв'язності і $m' - 1$ ребро. За індуктивним припущенням маємо $m' - 1 \geq n - (k + 1)$, або $m' \geq n - k$, що й потрібно було довести.

При доведенні справедливості верхньої оцінки можна вважати, що кожен компонент зв'язності графа G є повним графом. Припустимо, що C_i і C_j — компоненти зв'язності відповідно з n_i і n_j вершинами, де $n_i \geq n_j > 1$. Якщо замінити C_i і C_j на повні графи з $n_i + 1$ і $n_j - 1$ вершинами, то загальна кількість вершин не зміниться, а число ребер збільшиться на деяку додатну величину:

$$\begin{aligned} 1/2\{((n_i + 1) \cdot n_i - n_i \cdot (n_i - 1)) - ((n_j - 1) \cdot n_j + (n_j - 1) \cdot (n_j - 2))\} &= \\ &= 1/2\{n_i^2 + n_i - n_i^2 + n_i - n_j + n_j + n_j - 3n_j + 2\} = \\ &= 1/2(2n_i - 2n_j + 2) = n_i - n_j + 1. \end{aligned}$$

Отже, для того щоб число ребер в графі G було максимальним при даних n і k , граф G повинен складатися з $k - 1$ ізольо-

ваної вершини і повного графа з $n-k+1$ вершиною. А звідси одразу випливає потрібна нерівність.

Теорема доведена.

Наслідок 4.3.2. Будь-який граф порядку n , який має більше $(n-1) \cdot (n-2)/2$ ребер, зв'язний.

Дійсно, в цьому випадку число компонентів зв'язності такого графа повинне бути строго менше двох. Отже, $k=1$, тобто граф зв'язний.

Друге питання, яке теж часто виникає, — наскільки сильно зв'язним є зв'язний граф. Інакше це питання можна сформулювати так: скільки потрібно вилучити ребер з графа, щоб він перестав бути зв'язним?

Для відповіді на дане питання введемо деякі означення загального характеру.

Множина ребер довільного графа G , яка називається *множиною, що розділяє граф G* , — це така множина його ребер, вилучення якої з графа G викликає збільшення числа його компонентів зв'язності. Якщо граф G зв'язний, то множиною ребер G , що розділяє граф G , називається така множина його ребер, вилучення якої з графа G веде до незв'язного графа.

Приклад 4.3.1

У наведеному на рис. 4.3.3 графі G множина ребер $\{e_4, e_5, e_7\}$ являє собою множину, що розділяє граф G .

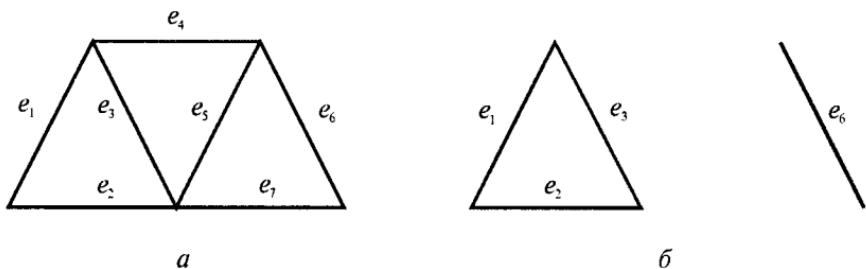


Рис. 4.3.3. Граф:

$a - G$; b — одержаний внаслідок вилучення множини ребер $\{e_4, e_5, e_7\}$

Очевидно, що $\{e_6, e_7\}$, $\{e_5, e_6, e_7\}$, $\{e_1, e_2, e_3\}$, $\{e_1, e_3, e_4\}$ — це множини ребер, які розділяють граф G . \blacktriangleleft

Отже, множина ребер, що розділяє граф, може мати власну підмножину, яка також розділяє граф.

Розрізом графа G називається така множина ребер, що розділяє граф G , жодна власна підмножина якої не є множиною, що

що розділяє цей граф. У розглянутому прикладі розрізами будуть множини $\{e_4, e_5, e_7\}$, $\{e_6, e_7\}$, $\{e_1, e_3, e_4\}$.

Розріз графа G , який складається лише з одного ребра, називається **мостом**. Наприклад, якщо граф G є колесом, то в ньому всякий його розріз буде мостом.

Зауважимо, що внаслідок вилучення ребер, які входять до складу розрізу графа G , число компонентів зв'язності графа G збільшується рівно на одиницю.

Граф, ізоморфний своєму доповненню, називається графом, що доповнює сам себе. Прикладами графів, які доповнюють самі себе, можуть служити графи: K_1 — повний граф порядку 1, C_5 — простий цикл довжини 5 і т.д. Очевидно також, що коли G ізоморфний H , то G^* ізоморфний H^* і $G^{**} = G$.

Теорема 4.3.7. Для будь-якого графа G або він сам, або його доповнення G^* є зв'язним.

Д о в е д е н н я. Нехай $G = (V, E)$ — незв'язний граф, A — одна із його компонентів зв'язності і $B = V \setminus A$. Тоді для будь-яких вершин u з A і v з B в графі G^* є маршрут довжини 1, оскільки ці вершини зв'язані ребром (u, v) . Отже, всяка вершина з B зв'язана з вершиною u маршрутом довжини 1, а всяка вершина з A — з u маршрутом довжини, не більше 2, тобто всяка вершина з A зв'язана з будь-якою вершиною v маршрутом. Значить, G^* — зв'язний граф.

Теорема доведена.

Теорема 4.3.8. Нехай $G = (V, E)$ — зв'язний граф і $e \in E$ — деяке його ребро. Тоді якщо e належить деякому циклу, то $G - e$ зв'язний. Якщо ж e не належить жодному циклу, то граф $G - e$ має рівно два компоненти зв'язності.

Д о в е д е н н я. Нехай $e = (u, v)$ належить деякому циклу C графа G . Замінimo в кожному ланцюкові, який з'єднує вершини x і y та містить ребро e , підланцюг (u, e, v) (u, v) -ланцюгом $C - e$. Одержано маршрут з вершини x у вершину y , який не має ребра e . Тобто в графі G всякі дві вершини, що не збігаються, з'єднані маршрутом. А це означає, що граф $G - e$ зв'язний.

Нехай тепер $e = (u, v)$ не входить у жодний цикл графа G . Тоді очевидно, що вершини u і v належать різним компонентам зв'язності, наприклад G_u і відповідно G_v графа $G - e$. Для довільної вершини $x \neq u$ і v в G існує маршрут із x в u внаслідок зв'язності G . Якщо e в цей маршрут не входить, то $x \in G_v$, тобто G_u і G_v — компоненти зв'язності.

Теорема доведена.

Ці теореми дають деяку характеристику операцій вилучення ребра щодо властивості зв'язності. Очевидно, що обернена

операція — операція введення ребра — не порушує цієї властивості.

5. МЕТРИЧНІ ХАРАКТЕРИСТИКИ ЗВ'ЯЗНИХ ГРАФІВ

Нехай $G = (V, E)$ — зв'язний граф, а u і v — дві його різні вершини.

Відстаню між вершинами u і v називається довжина найкоротшого маршруту, який з'єднує вершини u і v , і позначається $d(u, v)$. Покладемо також, що $d(u, u) = 0$. Очевидно, що для введеного таким чином відстані виконуються такі аксіоми:

- 1) $d(u, v) \geq 0$;
- 2) $d(u, v) = 0$ тоді і тільки тоді, коли $u = v$;
- 3) $d(u, v) = d(v, u)$;
- 4) $d(u, v) + d(v, w) \geq d(u, w)$ (нерівність трикутника).

Поняття відстані між вершинами дає можливість дати означення поняття степеня графа. Нехай G — зв'язний граф, k — натуральне число. Граф G^k — k -й степінь графа G , має ту ж множину вершин, що й G , а нерівні між собою вершини u і v суміжні в графі G^k тоді і тільки тоді, коли $d(u, v) \leq k$. Безпосередньо з означення випливає така властивість графа G^k :

Твердження 4.3.5. Якщо $k \geq |V| - 1$, де V — множина вершин графа G , то G^k — повний граф.

Нехай u — деяка фіксована вершина графа $G = (V, E)$. Величина $e(u) = \max_{v \in V} d(u, v)$, називається **екскентриситетом** вершини u . Максимальний серед всіх екскентриситетів вершин графа G називається **діаметром графа** G і позначається $d(G)$. Отже, $d(G) = \max_{u \in V} e(u)$.

Вершина v графа G називається **периферійною**, якщо $e(v) = d(G)$. Простий ланцюг довжини $d(G)$, відстань між початковою і кінцевою вершинами якого дорівнює $d(G)$, називається **діаметральним ланцюгом**.

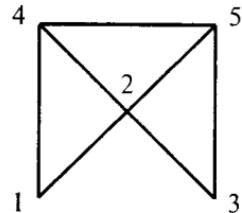


Рис. 4.3.4.
Граф G

Приклад 4.3.2

У даному графі G на рис. 4.3.4 маємо $d(1, 2) = 1$, $d(1, 3) = 2 = d(1, 5)$, $e(1) = 2$, $e(2) = 1$, $d(G) = 2$. Всі вершини, крім вершини 2, периферійні, а $(1, 2, 3)$ — діаметральний ланцюг. *

6. ВЛАСТИВОСТІ ЕЙЛЕРОВИХ ГРАФІВ

Одним з важливих різновидів зв'язних графів є так звані ейлерові і гамільтонові графи.

Зв'язний граф G називається *ейлеровим*, якщо існує замкнений ланцюг, який включає кожне його ребро. Такий ланцюг називається *ейлеровим ланцюгом*.

Зв'язний граф G називається *напівейлеровим*, якщо в ньому існує ланцюг, який включає кожне його ребро. Таким чином, всякий ейлерів граф буде напівейлеровим.

Приклад 4.3.3

На рис. 4.3.5 наведені графи: неейлерів (a), ейлерів (b) і напівейлерів (c).

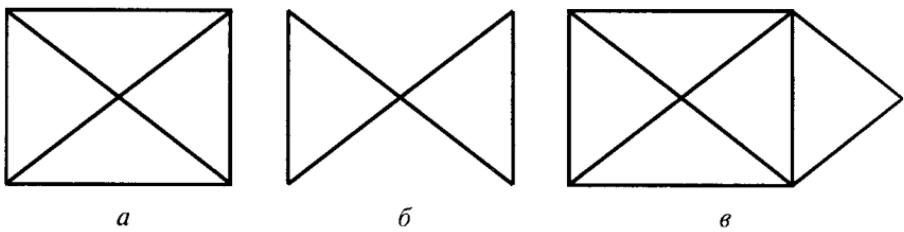


Рис. 4.3.5. Різновиди графів.

Поняття ейлерового графа виникло у зв'язку з відомою головоломкою про кенігсбергські мости, в якій необхідно було визначити, чи має граф, зображений на рис. 4.3.6, ейлерів ланцюг, чи ні.

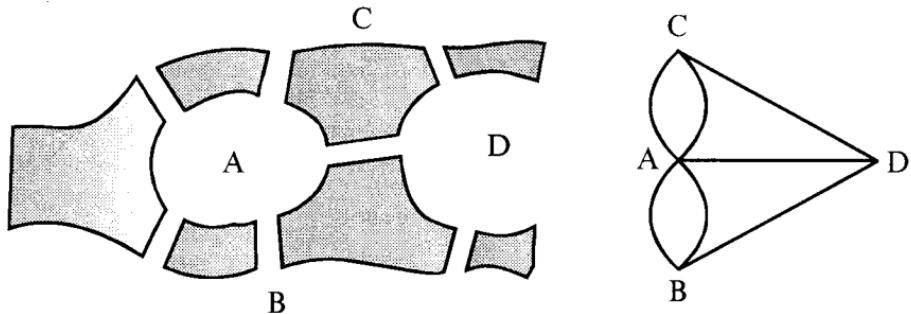


Рис. 4.3.6. Кенігсбергські мости

Виникає природне питання: як встановити, що заданий граф є ейлеровим? Відповідь на це питання дають такі твердження.

Теорема 4.3.9. Якщо степінь кожної вершини скінченного графа не менший двох, то граф має цикл.

Доведення. Нехай v — довільна вершина графа G . Користуючись методом побудови за індукцією, побудуємо маршрут $v, v_1, \dots, v_k, \dots$ так, що v_{i+1} суміжна з v_i і відмінна від v_{i-1} . Існування такої вершини випливає з умов теореми. Оскільки граф G скічений, то на деякому кроці побудови нашого маршруту прийдемо до вершини, вибраної раніше. Нехай v_k — перша з таких вершин. Тоді частина маршруту, яка лежить між першою і другою появою v_k в цьому маршруті, і є шуканим циклом.

Теорема доведена.

Зауважимо, що дана теорема справедлива для псевдографа (який має петлі) і для мультиграфа (який має кратні ребра).

Теорема 4.3.10. Зв'язний граф G є ейлеровим тоді і тільки тоді, коли кожна вершина G має парний степінь.

Доведення. Нехай G має ейлерів ланцюг P ; тоді при всякому проходженні ланцюга P через будь-яку з вершин графа G степінь цієї вершини збільшується на 2. А оскільки кожне ребро зустрічається в P тільки один раз, то кожна вершина повинна мати парний степінь.

Доведення у зворотному напрямку проводиться індукцією за числом ребер у графі G .

Наслідок 4.3.3. Зв'язний граф ейлерів тоді і тільки тоді, коли множину його ребер можна розбити на цикли, що не перетинаються.

Наслідок 4.3.4. Зв'язний граф напівейлерів тоді і тільки тоді, коли він має не більше двох вершин непарного степеня.

Зауважимо, що теорема справедлива для псевдографів (які мають 4 петлі) і для мультиграфів (які мають кратні ребра).

Слід також зауважити, що коли напівейлерів граф має рівно дві вершини непарного степеня, то у всякого напівейлерового ланцюга обов'язково одна з цих вершин початкова, а друга кінцева. З леми про рукостискання випливає, що граф не може мати тільки одну вершину непарного степеня.

Теорема Флері. Нехай $G = (V, E)$ — ейлерів граф. Тоді наступна процедура завжди можлива і веде до ейлерового ланцюга в графі G : виходячи з будь-якої вершини $u \in V$, ідемо по ребрах графа G довільним чином згідно з такими правилами:

(i) стираємо ребра, які пройдені, і стираємо ізольовані вершини, які при цьому виникають;

(ii) на кожному етапі ідемо по мосту лише тоді, коли немає інших можливостей.

Доведення. Насамперед покажемо, що вказану процедуру можна виконувати на кожному етапі. Нехай у процесі роботи процедури досягнута вершина v . Тоді якщо $v \neq u$, то підграф H , який залишився, зв'язний і має рівно дві вершини непарного

степеня, — це вершини u і v . За наслідком 4.3.4 граф H має напівейлерів ланцюг P із v в u . Оскільки вилучення непарного ребра ланцюга P не порушує зв'язності графа H , то звідси випливає, що побудова, яка дається в теоремі, завжди можлива на кожному етапі. Якщо ж $u = v$, то доведення залишається таким же доти, доки ще є ребра, інцидентні вершині u .

Залишається лише показати, що дана процедура завжди приводить до повного ейлерового ланцюга. Але це очевидно, оскільки в G не може бути ребер, які не були пройдені після використання останнього ребра, інцидентного вершині u (інакше, вилучення деякого ребра, суміжного з одним із тих, які залишилися, веде до незв'язного графа, а це суперечить умові (ii) теореми).

7. ВЛАСТИВОСТІ ГАМІЛЬТОНОВИХ ГРАФІВ

Розглянута вище проблема існування замкнутого ланцюга, який проходить через кожне ребро заданого зв'язного графа G , аналогічно може бути сформульована і для вершин. Тобто, чи існує замкнений ланцюг у заданому зв'язному графі G , який проходить рівно один раз через кожну вершину графа G . Очевидно, що такий ланцюг має бути циклом. Якщо такий цикл у графі G існує, то він називається **гамільтоновим циклом**, а граф G — **гамільтоновим графом**.

Пошук критерія гамільтоновості графа — це одна з основних невирішених проблем теорії графів. Про гамільтонові графи відомо ще зовсім мало. Один із найбільш відомих результатів загального характеру подається в такій теоремі.

Теорема Дірака. Якщо в графі $G = (V, E)$ з $n \geq 3$ вершинами ($v \in V$) $n(v) \geq n / 2$, то граф G є гамільтоновим.

Д о в е д е н н я. Введемо в граф G k нових вершин і з'єднаємо кожну з цих вершин із кожною вершиною графа G . Будемо вважати, що k — найменше число вершин, які потрібно ввести в граф G для того, щоб одержаний граф G' став гамільтоновим.

Припустимо, що $k > 0$ і v, p, w, \dots, v — гамільтонів цикл, де v і w — вершини з G , а p — одна з нових вершин. Тоді v і w несуміжні, оскільки в протилежному випадку вершина p зайва, а це суперечить мінімальності k . Більше того, вершина (наприклад, w'), суміжна з вершиною w , не може безпосередньо йти за вершиною v' , суміжною з v , оскільки тоді ми могли б замінити v, p, w, \dots, v' , w', \dots, v на $v, v', \dots, w, w', \dots, v$, перевернувши частину циклу, яка знаходиться між вершинами w і v' . Звідси випливає,

що число вершин графа G' , не суміжних з w , не менше числа вершин, суміжних з v , тобто, у крайньому випадку, воно дорівнює $n / 2 + k$. З іншого боку, очевидно, що число вершин графа G' , суміжних з w , теж не менше $n / 2 + k$. Оскільки всяка вершина графа G' не може бути одночасно суміжною і несуміжною з w , то загальна кількість вершин графа G' , яка дорівнює $n + k$, не менша $n + 2k$. Це може бути лише тоді, коли $k = 0$.

Одержана суперечність доводить теорему.

Якщо в графі $G = (V, E)$ порядку n зафіксувати одну з вершин і обхід графа завжди починати з неї, то всікому гамільтоновому циклу буде відповідати перестановка елементів множини V . Отже, знайти гамільтонів цикл або переконатися в його відсутності можна перебором $(n - 1)!$ перестановок. Якщо граф G гамільтонів, то цей перебір у повному обсязі необхідно буде зробити лише у випадку великого невезіння, тобто коли перестановка, що відповідає гамільтоновому циклу, зустрічається в цьому процесі останньою. Якщо ж граф G — негамільтонів, то, діючи таким чином, необхідно буде перебрати всі $(n - 1)!$ перестановку. Хоча на практиці користуються різними алгоритмами часткового перебору, але складність цих алгоритмів залишається високою (пропорційно $(n - 1)!$).

§ 4.4. МАТРИЦІ І ГРАФИ

1. МАТРИЦІ СУМІЖНОСТЕЙ І ДОСЯЖНОСТІ

З розділу 1 (див. § 1.1) ми знаємо, що всяке бінарне відношення можна задати у вигляді матриці, яка складається з нулів і одиниць і називається **матрицею суміжностей**. Взаємно однозначна відповідність між бінарними відношеннями і графами на деякій скінченній множині дає можливість стверджувати, що всякий граф можна подавати у вигляді матриці суміжностей. Дійсно, якщо $G = (V, E)$ — скінчений граф порядку n , то йому відповідає квадратна матриця $A(G)$ розмірності $n \times n$, яка має вигляд

$$a_{ij} = \begin{cases} 1, & \text{коли вершини } v_i \text{ і } u_j \text{ суміжні,} \\ 0 & \text{— в протилежному випадку.} \end{cases}$$

Внаслідок симетричності відношення суміжності матриця суміжностей графа має бути симетричною, а число одиниць в i -му рядку — дорівнювати степеню вершини u_i .

Задання графів за допомогою матриць суміжностей дозволяє сформулювати простий критерій ізоморфності графів.

Теорема 4.4.1. Графи ізоморфні тоді і тільки тоді, коли їх матриці суміжностей можна одержати одну з другої однаковими перестановками рядків і стовпчиків.

Д о в е д е н н я. Перенумеруємо вершини графів $G = (V, E)$ і $H = (V_1, E_1)$ ($|V| = |V_1| = n$) цілыми числами від 1 до n .

Якщо $A(G) = B(H)$, то все доведено. У протилежному випадку графи G і H відрізняються лише нумерацією вершин. Значить, існує така підстановка s на множині вершин V , яка зберігає суміжність, тобто якщо $(u, v) \in E$, то $(s(u), s(v)) \in E_1$. Тоді маємо $b_{s(i)} s(j) = a_{ij}$, $i, j = 1, 2, \dots, n$.

Теорема доведена.

З наведеної теореми видно, що представлення графа матрицею суміжностей дає важливу інформацію про природу графа. Пам'ятаючи про те, що коли $A(R)$ — матриця суміжностей відношення R , то $A(R)^2$ — матриця суміжностей відношення R^2 , $A(R)^3$ — матриця суміжностей відношення R^3 і т.д., ми приходимо до такого твердження.

Теорема 4.4.2. Якщо A — матриця суміжностей графа G , який має порядок n , то елемент b_{ij} матриці A^k є числом маршрутів довжини k , які з'єднують вершини u_i та u_j в графі G .

Д о в е д е н н я проводиться індукцією за числом k .

Для $k = 1$ теорема очевидна, оскільки маршрут довжини 1 є ребром графа G .

Нехай теорема справедлива для всіх $k < m$. Покажемо, що вона справедлива і для $k = m$.

Нехай b_{iq} — елемент матриці A^{m-1} , a_{qj} — елемент матриці A . Тоді згідно з припущенням індукції $b_{iq}a_{qj}$ є число маршрутів довжини m , які з'єднують вершини u_i та u_j і проходять через вершину u_q . Значить, сума

$$\sum_{q=1}^n b_{iq}a_{qj} \quad (1)$$

є числом маршрутів довжини m , які з'єднують вершини u_i та u_j в графі.

Теорема доведена.

Наслідок 4.4.1. В графі порядку n маршрут, що з'єднує вершини u і v , існує тоді і тільки тоді, коли елемент матриці

$$C = \sum_{k=1}^n A^k, \text{, що відповідає цим вершинам, не дорівнює нулю.}$$

Д о в е д е н н я. Якщо відповідний елемент $c = 0$, то за теоремою 4.4.2 та згідно з твердженнями 4.3.1 і 4.3.2 існує ланцюг, який з'єднує вершини u , u_1, \dots, v , довжиною не більше, ніж n . Якщо маршрут, який з'єднує u і v , існує, то згідно з твердженнями 4.3.1 і 4.3.2 існує маршрут довжиною $k < n$, який з'єднує ці

вершини. За теоремою 4.4.2 в матриці C відповідний елемент c не дорівнює нулю, що і потрібно було довести.

Зауважимо, що в деяких випадках при розв'язуванні задач на графах знаходження матриці C і степенів матриці A можна значно спростити, якщо при обчисленні суми (1) використовувати булеві зв'язки \vee , $\&$, тобто обчислювати $\bigvee_{q=1}^n a_{iq} a_{qj}$. У цьому разі елементи матриць спрощуються і обчислення виконуються ефективніше. Одержану таким чином матрицю C будемо називати **матрицею досяжності графа**.

Наслідок 4.4.2. Граф зв'язний тоді і тільки тоді, коли кожний елемент матриці досяжності графа дорівнює одиниці.

Аналогічно визначається матриця суміжностей орграфа $G = (V, E)$:

$$a_{ij} = \begin{cases} 1, & \text{якщо } (u_i, u_j) \in E, \\ 0 & \text{в протилежному випадку.} \end{cases}$$

Матриці суміжностей неоріентованого графа і орграфа розрізняються тим, що перша завжди симетрична, а друга не обов'язково симетрична. Очевидно, що всяка квадратна матриця, елементи якої 0 і 1, буде матрицею суміжностей деякого оріентованого графа.

Приклад 4.4.1

Нехай маємо матрицю

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

Цій матриці відповідає орграф, зображеній на рис. 4.4.1.

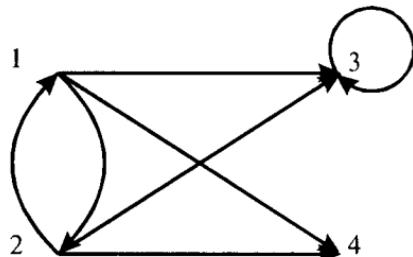


Рис. 4.4.1. Орграф

Із теореми 4.4.1 про ізоморфізм графів випливає, що ранги матриць суміжностей ізоморфних графів рівні між собою. Це дає можливість ввести таке означення.

Рангом графа G називається ранг його матриці суміжностей, який будемо позначати $\text{rank}(G)$.

2. МАТРИЦЯ КІРХГОФА

Нехай $G = (V, E)$ порядку n і $V = \{1, 2, \dots, n\}$. Дамо означення матриці $B(G)$ для графа G таким чином:

$$b_{ij} = \begin{cases} -1, & \text{якщо } (i, j) \in E; \\ 0, & \text{якщо } i \neq j \text{ і } (i, j) \notin E; \\ n(i), & \text{якщо } i = j. \end{cases}$$

Матриця $B(G)$ називається *матрицею Кірхгофа* графа G . Сума елементів кожного рядка і кожного стовпчика цієї матриці дорівнює нулю.

Зауважимо, що теорема про ізоморфізм графів залишається справедливою, якщо розглядати не матриці суміжностей, а матриці Кірхгофа. Доведення теж повністю зберігається.

Теорема 4.4.3. *Нехай B — довільна числова квадратна матриця розмірності n , в кожному рядку і стовпчику якої сума елементів дорівнює нулю, тобто*

$$\sum_{j=1}^n b_{ij} = 0 \quad (i = 1, 2, \dots, n), \quad \sum_{i=1}^n b_{ij} = 0 \quad (j = 1, 2, \dots, n).$$

Тоді алгебраїчні доповнення всіх елементів матриці B рівні між собою. Зокрема, цю властивість має матриця Кірхгофа довільного графа.

Д о в е д е н н я. Очевидно, що $\text{rank}(B) < n$. Якщо $\text{rank}(B) < n - 1$, то алгебраїчні доповнення всіх елементів цієї матриці дорівнюють нулю. Нехай $\text{rank}(B) = n - 1$ і C — матриця, елементами якої є c_{ij} , що являють собою алгебраїчні доповнення елемента b_{ij} і матриці B , $i = 1, 2, \dots, n, j = 1, \dots, n$. Відомо, що $BC = |B|E$, де $|B|$ — визначник матриці B , а E — одинична матриця. Значить, $BC = 0$, оскільки $|B| = 0$ ($\text{rank}(B) = n - 1$). Отже, для стовпчика матриці C з номером j ($j = 1, 2, \dots, n$) справедливі рівності $b_{1j}c_{1j} + b_{2j}c_{2j} + \dots + b_{nj}c_{nj} = 0$, $i = 1, 2, \dots, n$. Їх можна розглядати як систему лінійних однорідних рівнянь з матрицею B відносно невідомих $b_{1j}, b_{2j}, \dots, b_{nj}$. Оскільки $\text{rank}(B) = n - 1$, то всі розв'язки цієї системи пропорційні. Крім того, вектор $(1, 1, \dots, 1)$ відповідає даній системі і тому $b_{1j} = b_{2j} = \dots = b_{nj}$, $j = 1, 2, \dots, n$. Враховуючи те, що і $C \cdot B = 0$, аналогічно одержуємо $b_{1j} = b_{2j} = \dots = b_{nj}$, $j = 1, 2, \dots, n$. Значить, $b_{ij} = b_{kl}$, $i, j, k, l = 1, 2, \dots, n$.

Теорема доведена.

3. МАТРИЦЯ ІНЦІДЕНТНОСТІ ГРАФА

Нехай $G = (V, E)$ — граф, де $V = \{1, 2, \dots, n\}$, $E = \{e_1, \dots, e_m\}$. Визначимо матрицю $I(G)$ такими умовами:

$$i_{kl} = \begin{cases} 1, & \text{якщо вершина } k \text{ і ребро } e_l \text{ інцидентні;} \\ 0 & \text{— в протилежному випадку.} \end{cases}$$

Матриця $I(G)$ називається матрицею інцидентності графа G .

Для орієнтованих графів означення матриці інцидентності $I(G)$ дещо змінюється:

$$i_{kl} = \begin{cases} 1, & \text{якщо вершина } k \text{ — початок дуги } e_l; \\ -1, & \text{якщо вершина } k \text{ — кінець дуги } e_l; \\ 0, & \text{якщо вершина } k \text{ і дуга } e_l \text{ не інцидентні.} \end{cases}$$

Для матриць інцидентності справедливе твердження, аналогічне теоремі 4.4.1.

Теорема 4.4.4. *Графи (орграфи) ізоморфні тоді і тільки тоді, коли їх матриці інцидентності можна одержати одну з другої довільними перестановками рядків і стовпчиків.*

Нехай G — граф. Граф G можна перетворити в орграф, якщо кожному ребру надати одну з двох можливих орієнтацій. Одержаній орграф називається *орієнтацією графа* G . Безпосередньо перевіркою встановлюється справедливість такого твердження.

Теорема 4.4.5. *Якщо B — матриця Кірхгофа графа G , а I — матриця інцидентності деякої його орієнтації H (вершини в H перенумеровані так, як і в G), то $B = I \cdot I'$, де I' — транспонована матриця I .*

§ 4.5. ПЛАНАРНІСТЬ І УКЛАДАННЯ ГРАФІВ

Розглянемо питання про те, чи всякий граф можна зобразити на площині.

1. ПЛОСКІ І ПЛАНАРНІ ГРАФИ

Плоским графом називається граф, вершини якого є точками площини, а ребра — неперервними плоскими лініями без самоперетинів, які з'єднують відповідні вершини так, що ніякі два ребра не мають спільних точок, крім інцидентної їм обом вершини.

Прикладами плоских графів можуть служити графи, зображені на рис. 4.5.1.

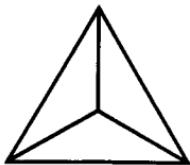
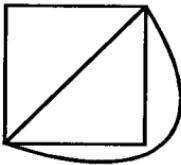
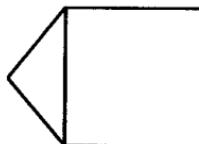
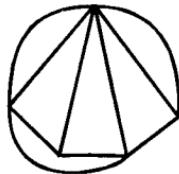
*a**b**c**d*

Рис. 4.5.1. Плоскі графи

Граф називається **планарним**, якщо він ізоморфний деякому плоскому графу. Граф K_4 , зображений на рис. 4.5.2, планарний, оскільки він ізоморфний графам *a* і *b*, зображенням на рис. 4.5.1.

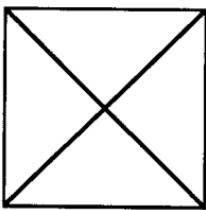


Рис. 4.5.2.

Планарний граф

жорданова крива, початок і кінець якої збігаються. Аналогічно можна дати означення жорданової кривої в тривимірному просторі або на таких поверхнях, як сфера, тор тощо. Знаменита теорема Жордана формулюється так.

Теорема Жордана. Якщо S — замкнута жорданова крива на площині, а x і y — дві різні точки цієї кривої, то всяка жорданова крива, яка з'єднує точки x і y , лежить або повністю всередині S , крім точок x і y , або поза кривою S , крім точок x і y , або перетинає криву S в деякій точці, відмінній від точок x і y (рис. 4.5.3).

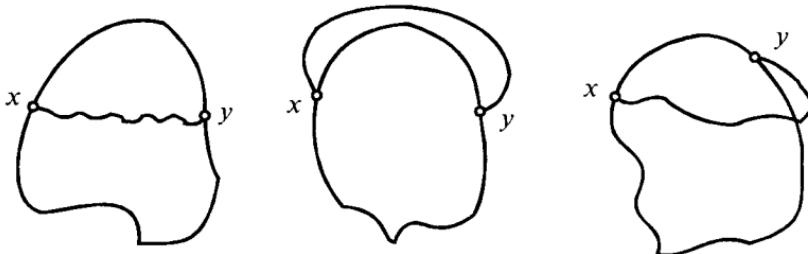


Рис. 4.5.3. Жорданова крива

Будемо говорити, що граф G укладається в просторі L , якщо існує взаємно однозначне відображення вершин і ребер графа G

відповідно в точки і жорданові криві цього простору таке, що криві, які відповідають різним ребрам, перетинаються в інцидентних цим ребрам вершинах. Зображеній таким чином граф G в просторі L називається **укладенням графа G** .

Теорема 4.5.1. *Всякий граф укладається в тривимірному просторі.*

Д о в е д е н н я. Розмістимо всі вершини графа $G = (V, E)$ на осі OX . Із пучка площин, які проходять через цю вісь, виберем $|E|$ різних площин. Далі кожне ребро $(u, v) \in E$ зобразимо у відповідній площині півколом, яке проходить через вершини u і v . Ясно, що в результаті одержимо укладення графа G в тривимірний простір, оскільки всі ребра лежать у різних площинах і тому не перетинаються в жодних точках, крім вершин.

Теорема доведена.

Теорема 4.5.2. *Граф укладається на сфері тоді і тільки тоді, коли він планарний.*

Д о в е д е н н я. Розглянемо стереографічну проекцію. Для цього проведемо площину Q , яка дотикається до сфери, так, щоб точка N , діаметрально протилежна точці дотику (північний полюс N), не збігалася з вершиною графа і не лежала на його ребрі. Нехай граф G укладений на сфері. Розглянемо граф G' , одержаний за допомогою стереографічної проекції графа G з точки N на площину Q . Оскільки відповідність між точками сфери, відмінними від N , та їх стереографічними проекціями взаємно однозначна, то граф G плоский і ізоморфний графу G' . Отже, граф G планарний.

Обернене твердження доводиться аналогічно з урахуванням взаємно однозначної відповідності, яка була встановлена.

Теорема доведена.

Слід зауважити, що означення плоского і планарного графів, так як і теореми 4.5.1, 4.5.2 та багато інших тверджень зберігаються для мультиграфів і псевдографів.

Перш ніж перейти до пошуку критерію планарності графа, розглянемо добре відому задачу-головоломку про три будинки і три колодязі. Є три будинки: 1, 2, 3 і три колодязі: 4, 5, 6 (рис. 4.5.4). Кожний мешканець може користуватися будь-яким із трьох колодязів. Одного дня мешканці будинків вирішили прокласти доріжки до колодязів так, щоб виключити можливість зустрічі, тобто щоб доріжки не перетиналися. Виникає питання: чи можна прокласти доріжки так, як цього вимагає задача? Всі спроби прокласти дев'ять доріжок так, щоб вони не перетиналися і з'єднували будинки з колодязями, закінчуються невдало. Виявляється, що це зовсім не випадково.

Наведена задача дає привід думати, що існують не тільки планарні графи. І це стверджується такою теоремою.

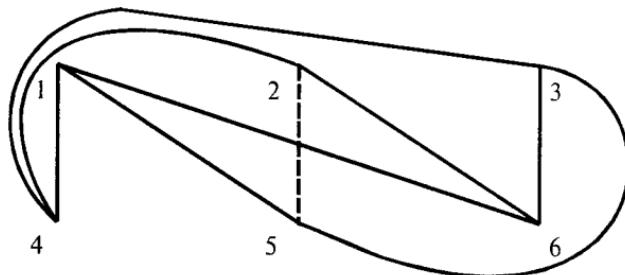


Рис. 4.5.4. Задача про три будинки і три колодязі

Теорема 4.5.3. *Графи K_5 і $K_{3,3}$ не є планарними.*

Доведення. Припустимо супротивне, що граф K_5 планарний. Оскільки він має цикл довжиною 5, наприклад v, w, x, y, z , то без обмеження загальності можна вважати, що при всякому укладенні цього графа на площині цей цикл зображується правильно п'ятикутником (рис. 4.5.5).

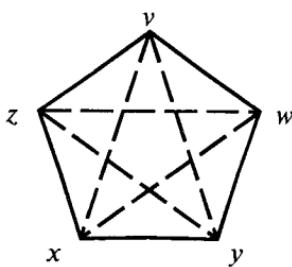


Рис. 4.5.5. Граф K_5

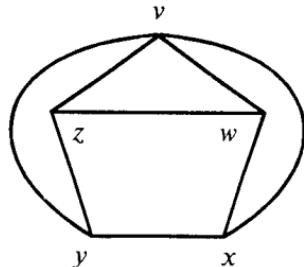


Рис. 4.5.6. Зображення укладення графа п'ятикутником

За теоремою Жордана ребро (z, w) повинно лежати або повністю всередині цього п'ятикутника, або повністю поза ним. Третю можливість (коли ребро має спільну точку з п'ятикутником) ми не розглядаємо, оскільки маємо справу з укладенням на площині.

Розглянемо спочатку випадок, коли (z, w) лежить всередині п'ятикутника. Оскільки ребра (v, x) і (v, y) не повинні перетинати ребро (z, w) , то обидва ці ребра лежать поза п'ятикутником: ця ситуація зображена на рис. 4.5.6. Ребро (x, z) не може перетинати ребро (v, y) , і тому воно повинно лежати всередині п'ятикутника. Аналогічно і ребро (w, y) теж повинно лежати всередині п'ятикутника. Але в цьому випадку ребра (w, y) і (x, z) обов'язково перетнуться, і ми одержуємо суперечність з нашим припущенням.

Другий випадок доводиться цілком аналогічно до розглянутого і пропонується як вправа.

Розглянемо тепер граф $K_{3,3}$. Припустимо, що він планарний. Тоді існує цикл довжини 6, наприклад u, v, w, x, y, z, u , який можна зобразити у вигляді шестикутника (рис. 4.5.7). Виконуючи доведення так, як це робилось вище для K_5 , приходимо до ситуації, коли двоє з ребер (u, x) , (v, y) , (w, z) повинні лежати або всередині шестикутника або поза шестикутником, а тому вони не можуть не перетинатися. Одержана суперечність повністю доводить теорему.

Безпосередньо з означення планарності графа випливають такі очевидні твердження.

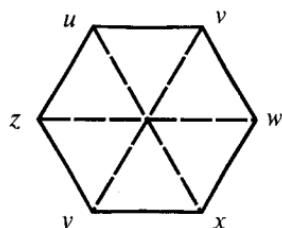


Рис. 4.5.7.
Граф $K_{3,3}$

Твердження 4.5.1. Всякий підграф планарного графа теж планарний.

Твердження 4.5.2. Якщо граф містить непланарний підграф, то і сам граф непланарний.

З останнього твердження випливає, що всякий граф, який містить K_5 і $K_{3,3}$ як підграфи, не буде планарним. Виявляється, що графи K_5 чи $K_{3,3}$ — єдині непланарні графи, оскільки всякий непланарний граф містить один із них як підграф. Для формулювання цього результату необхідно ввести поняття *гомеоморфності графів*.

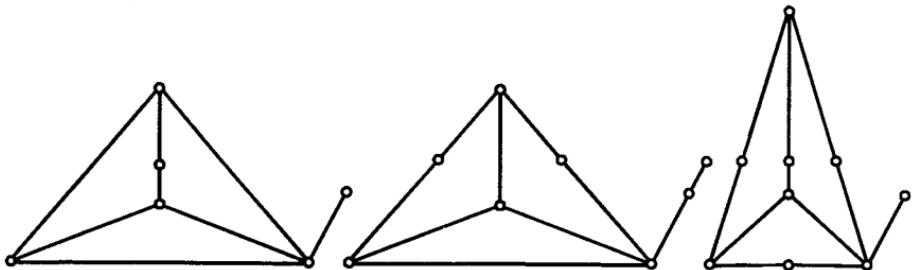


Рис. 4.5.8. Гомеоморфні графи

Два графи називаються *гомеоморфними* або *тотожними* з точністю до вершин степеня 2, якщо вони можуть бути одержані з одного і того ж графа за допомогою операції введення вершини в ребро степеня 2. Наприклад, графи, зображені на рис. 4.5.8, гомеоморфні.

Неважко переконатися, що гомеоморфізм графів є відношенням еквівалентності.

Теорема Понtryгіна—Куратовського. Граф планарний тоді і тільки тоді, коли він не має підграфів, гомеоморфних K_5 або $K_{3,3}$.

Доведення цієї теореми не наводиться, оскільки воно досить складне і вимагає розгляду деяких понять, які виходять за рамки даного підручника. Для компенсації цього, виходячи з теореми Понtryгіна—Куратовського, розглянемо інший критерій планарності графів.

Теорема 4.5.4. Граф планарний тоді і тільки тоді, коли він не має підграфів, які стягаються до графів K_5 і $K_{3,3}$.

Д о в е д е н н я. Припустимо, що граф G непланарний. Тоді за теоремою Понtryгіна—Куратовського він має підграф H , гомеоморфний K_5 чи $K_{3,3}$. Стягуючи ребра з H , інцидентні вершинам степеня 2, бачимо, що H стягається до K_5 або до $K_{3,3}$.

Нехай тепер граф G містить підграф H , який стягається до $K_{3,3}$, і вершина v графа $K_{3,3}$ одержана за допомогою операції стягування підграфа H_v до графа H (рис. 4.5.9).

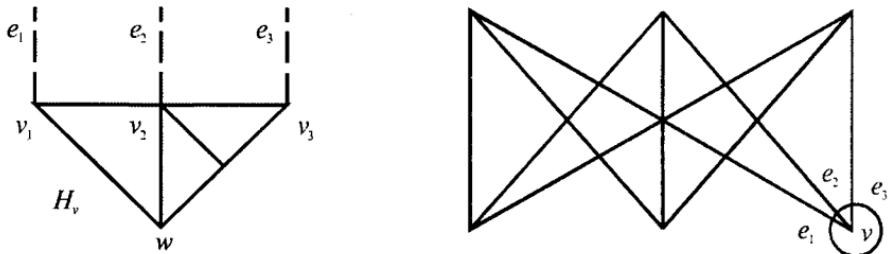


Рис. 4.5.9. Стягування підграфа H_v до графа H

В графі $K_{3,3}$ вершина v інцидентна трьом ребрам: e_1 , e_2 і e_3 ; як ребра з H вони інцидентні трьом (не обов'язково різним) вершинам: v_1 , v_2 , v_3 підграфа H_v . Якщо v_1 , v_2 , v_3 різні, то можна знайти вершину w із H_v і три простих ланцюги, що ведуть із w в ці вершини і перетинаються тільки у вершині w . Можна виконати аналогічні побудови і в тому випадку, коли не всі вершини різні; при цьому прості ланцюги перетворюються в окремі вершини. Отже, підграф H_v можна замінити вершиною w з трьома простими ланцюгами, що виходять із цієї вершини. Якщо такі побудови виконати для кожної вершини графа $K_{3,3}$, а одержані прості ланцюги доповнити відповідними ребрами з $K_{3,3}$, то ясно, що одержаний підграф буде гомеоморфним графу $K_{3,3}$. За теоремою Понtryгіна—Куратовського граф G непланарний (рис. 4.5.10).

Аналогічно доводиться і випадок, коли G включає підграф, який стягається до графа K_5 .

Теорема доведена.

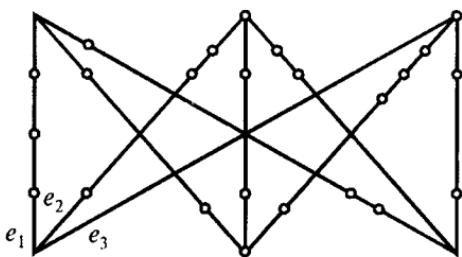


Рис. 4.5.10. Непланарний граф

2. ГРАНІ ПЛОСКОГО ГРАФА. ФОРМУЛА ЕЙЛЕРА

У цьому пункті встановимо формулу Ейлера, яка зв'язує число вершин, ребер і граней зв'язного плоского графа. Дамо спочатку означення поняття *грані плоского графа* G .

Точка x площини S , на якій розміщений граф G , називається *диз'юнктною з G* точкою, якщо точка x не відповідає жодній вершині графа G і не лежить на жодному ребрі цього графа. Дві точки площини S x і y називаються еквівалентними, якщо вони диз'юнктні з G і їх можна з'єднати такою жордановою кривою, всі точки якої диз'юнктні з G (рис. 4.5.11). Введене відношення еквівалентності точок площини розбиває множину всіх точок площини S на класи еквівалентності, які і називаються *гранями графа G* . Наприклад, граф G , зображений на рис. 4.5.12, має чотири грані: f_1, f_2, f_3, f_4 , і грань f_4 — нескінчена його грань.

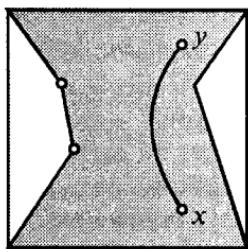


Рис. 4.5.11.
Диз'юнктні з G точки

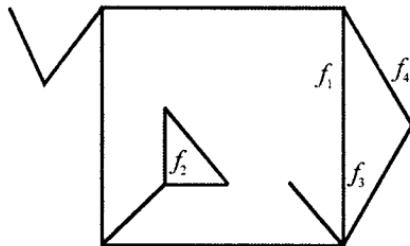


Рис. 4.5.12.
Граф з нескінченною гранню

Теорема Ейлера. *Нехай G — зв'язний плоский граф, в якому n , m і f — відповідно число вершин, ребер і граней. Тоді справедлива рівність $n + f = m + 2$.*

Доведення виконується індукцією за числом ребер у графі G .

Якщо $m = 0$, то $n = 1$ (оскільки G зв'язний) і $f = 1$ (некінчена грань), тобто теорема справедлива в цьому випадку.

Нехай теорема справедлива для довільного графа G , який має $m - 1$ ребро. Додамо до G нове ребро e (зауважимо, що це не є операцією введення ребра в граф). Тоді можливі такі випадки:

а) ребро e є петлею, і в цьому випадку число граней збільшується на одну, а число вершин залишається незмінним;

б) ребро e з'єднує дві різні вершини в G , і в цьому випадку одна з граней графа G розпадається на дві, число граней зростає на одиницю, а число вершин залишається незмінним;

в) ребро e інцидентне лише одній із вершин графа G , і в цьому випадку число його вершин зростає на одиницю, а число граней залишається незмінним.

У кожному з цих випадків теорема залишається справедливою, а оскільки наведеними випадками вичерпуються всі можливості, то цим теорема доведена повністю.

Теорему Ейлера легко перенести і на незв'язні графи.

Наслідок 4.5.1. Нехай G — плоский граф з n вершинами, m ребрами, f гранями і k компонентами зв'язності; тоді $n + f = m + k + 1$.

Д о в е д е н н я. Результат одержуємо, застосувавши теорему Ейлера окремо до кожного з компонентів. При цьому нескінчenna грань враховується лише один раз.

Наслідок 4.5.2. Якщо G — зв'язний планарний граф з m ребрами і $n \geq 3$ вершинами, то $m \leq 3n - 6$.

Д о в е д е н н я. Оскільки кожна грань графа обмежена не менше ніж трьома ребрами, то підраховуючи число ребер навколоожної з гранеї, одержуємо, що $3f \leq 2m$. Множник 2 з'являється тому, що кожне ребро обмежує не більше двох граней. Отже, маємо $3(m + 2) = 3m + 6 = 3(n + f) \leq 3n + 2m$, тобто $m \leq 3n - 6$.

Наслідок 4.5.3. Графи $K_{3,3}$ і K_5 не є планарними.

Д о в е д е н н я. Якби K_5 був планарним, то за наслідком 4.5.2 мали б, що $10 \leq 9$. А це очевидна суперечність.

Для непланарності графа $K_{3,3}$ достатньо зауважити, що кожна його грань обмежена щонайменше чотирма ребрами і, значить, $4f \leq 2m$, або $2f \leq 9$. Але цього не може бути, оскільки з теореми Ейлера випливає, що $f = 5$.

Наслідок 4.5.4. У всякому планарному графі існує вершина, степінь якої не більше п'яти.

Д о в е д е н н я. Без обмеження загальності можна вважати, що граф плоский, зв'язний і має щонайменше три вершини. Якщо степінь кожної з його вершин не менший шести, то маємо $6n \leq 2m$, або $3n \leq m$. Згідно з наслідком 4.5.2 $3n \leq 3n - 6$ — очевидна суперечність.

§ 4.6. РОЗФАРБУВАННЯ ГРАФІВ. ХРОМАТИЧНЕ ЧИСЛО

Задачі розфарбування вершин чи ребер графа займають важливе місце в теорії графів. До задачі побудови розфарбування графа зводиться цілий ряд практичних задач.

1. ПРАВИЛЬНЕ РОЗФАРБУВАННЯ

Нехай $G = (V, E)$ — скінчений граф, а k — деяке натуральне число. Довільна функція $f: V \rightarrow N_k$, де $N_k = \{1, 2, \dots, k\}$, називається **вершинним k -розфарбуванням**, або просто **k -розфарбуванням** графа G . Розфарбування називається **правильним**, якщо

$$(\forall (u, v) \in E) (f(u) \neq f(v)).$$

Граф, для якого існує правильне k -розфарбування, називається **розфарбованим графом**. При визначені розфарбування графа множину N_k можна замінити довільною k -елементною множиною.

Правильне розфарбування графа можна трактувати як розфарбуванняожної його вершини в один із k кольорів, таким чином, щоб суміжні вершини були розфарбовані в різні кольори. Оскільки функція f не обов'язково взаємно однозначна, то при k -розфарбуванні фактично може бути використано менше ніж k кольорів. Отже, правильне k -розфарбування можна розглядати як розбиття множини вершин V графа G на класи $V_1 \cup V_2 \cup \dots \cup V_l = V$, де $l \leq k$, $V_i \neq \emptyset$, $i = 1, 2, \dots, l$. Кожний клас V_i — незалежна множина, а самі класи називаються **колірними класами**.

Мінімальне число k , при якому існує правильне k -розфарбування графа G , називається **хроматичним числом** цього графа і позначається $X_p(G)$. Якщо $X_p(G) = k$, то граф G називається **k -хроматичним**. Правильне k -розфарбування графа G при $k = X_p(G)$ називається **мінімальним**.

Приклад 4.6.1

Розглянемо граф G , зображений на рис. 4.6.1, де показано одне із правильних k -розфарбувань. Натуральними числами 1, 2, 3, 4 позначені фарби відповідних вершин. ▲

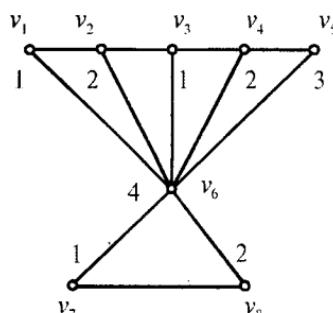


Рис. 4.6.1

2. ПРАКТИЧНІ ЗАДАЧІ, ЩО ЗВОДЯТЬСЯ ДО ЗАДАЧІ РОЗФАРБУВАННЯ

Задача складання розкладу занять. Нехай потрібно прочитати кілька лекцій за найкоротший проміжок часу. Читання лекцій займає одну годину, але деякі лекції не можуть читатися одночасно (наприклад, коли їх читає один і той же лектор). Побудуємо граф $G = (V, E)$, де V — множина, яка відповідає множині лекцій, причому дві вершини суміжні тоді і тільки тоді, коли відповідні лекції не можуть читатися одночасно. Очевидно, всяке правильне розфарбування цього графа визначає допустимий розклад: лекції, які відповідають вершинам графа, що становлять один колірний клас, читаються одночасно. Навпаки, всякий допустимий розклад визначає правильне розфарбування графа G . Оптимальний розклад відповідає мінімальним розфарбуванням, а число годин, необхідне для того, щоб прочитати всі лекції, дорівнює $X_p(G)$.

Задача розподілу устаткування. Дано множини $V = \{v_1, \dots, v_n\}$ і $S = \{s_1, s_2, \dots, s_m\}$ — види робіт і механізмів відповідно. Для виконанняожної з робіт необхідний певний час, однаковий для всіх робіт, і деякі механізми. При цьому жодний із механізмів не може одночасно використовуватися в кількох роботах. Необхідно розподілити механізми так, щоб загальний час виконання всіх робіт був мінімальним.

Побудуємо граф $G = (V, E)$, де $V = \{v_1, \dots, v_m\}$, а $(v_i, v_j) \in E$ тоді і тільки тоді, коли для виконання робіт v_i і v_j потрібен хоча б один спільний механізм. При правильному розфарбуванні графа G роботи, які відповідають вершинам одного кольору, можна виконувати одночасно, а найменший час виконання всіх робіт досягається при мінімальному розфарбуванні.

Задача проектування коробки швидкостей [29]. Коробка швидкостей — це механізм для зміни частоти обертів валу, який регулює швидкість руху автомобіля. Ця зміна виконується за рахунок того, що шестірні, які знаходяться всередині коробки, зчіплюються спеціальним способом. Одна із задач, яка стоїть перед конструктором коробки, полягає в мінімізації її розмірів, а це часто зводиться до мінімізації числа валів, на яких розміщені шестірні.

Побудуємо граф $G = (V, E)$, де V — множина шестірень і $(u, v) \in E$, якщо шестірні u і v не можуть бути розміщені на одному валу (наприклад, вони повинні зчіплюватися, чи вони занадто важкі для одного валу). Вершини, які входять до одного колірного класу, при правильному розфарбуванні цього графа відповідають шестірням, які можуть розміщуватися на одному

валу, а хроматичне число $X_p(C)$ дорівнює мінімальній кількості валів, необхідних для коробки, що проектується.

3. ХРОМАТИЧНІ ЧИСЛА ДЕЯКИХ ГРАФІВ

Для деяких простих графів неважко знайти хроматичні числа. Наприклад,

$$X_p(K_n) = n, X_p(K_n - e) = n - 1, X_p(K_{m,n}) = 2, \\ X_p(C_{2n}) = 2, X_p(C_{2n+1}) = 3.$$

Очевидно, що граф є 1-хроматичним тоді і тільки тоді, коли він пустий, а 2-хроматичним — коли він двочастинний і непустий. 2-Хроматичні графи, як правило, називають *біхроматичними*. Тому теорему Кьоніга про двочастинні графи можна сформулювати так.

Теорема 4.6.1. *Непустий граф є біхроматичним тоді і тільки тоді, коли він не має циклів непарної довжини.*

При яких умовах граф є 3-хроматичним, невідомо, хоча легко знайти приклади таких графів. До них належить граф Петерсена, циклічні графи з непарним числом вершин, а також колеса з непарним числом вершин. Взагалі про хроматичне число графа мало що можна сказати. Очевидно, що коли граф має n вершин, то його хроматичне число не перевищує n , а коли граф має підграф K_m , то його хроматичне число не менше m . Значно більшу інформацію про хроматичне число графа можна одержати, якщо відомі степені кожної його вершини.

Нехай r означає максимальний степінь вершини графа $G = (V, E)$.

Теорема 4.6.2. *Для всякого графа G виконується нерівність $X_p(G) \leq r + 1$.*

Доведення виконується індукцією за числом n вершин графа G .

Якщо $n \leq r + 1$, то теорема справедлива. Нехай теорема справедлива для всіх $n < k$. Покажемо її справедливість для $n = k$.

Якщо в графі G вилучити довільну вершину v , то в новому графі буде $n - 1$ вершина, і степінь будь-якої вершини цього графа не більший, ніж r . За припущенням індукції цей граф розфарбовується $r + 1$ кольором. Але звідси легко знайти розфарбування для графа G , якщо пофарбувати вершину v кольором, відмінним від кольорів вершин, суміжних з вершиною v . Оскільки цих вершин не більше r , то G розфарбовується $r + 1$ фарбою.

Теорема доведена.

Наслідок 4.6.1. Всякий кубічний граф розфарбовується чотирма фарбами.

Більш загальний результат, ніж теорема 4.6.2, дає така теорема.

Теорема Брукса. Якщо G — зв'язний неповний граф і $r \geq 3$, то $X_p(G) \leq r$.

Доведення цієї теореми опускається (його можна знайти, наприклад, в [16, 48]).

Хоча обидві теореми і дають певну інформацію про хроматичне число графа, але їх оцінки досить неточні. Дійсно, зірковий граф $K_{1,n}$, який за теоремою Брукса розфарбовується n фарбами, насправді є біхроматичним. Ця ситуація значно спрощується, якщо обмежитися планарними графами. У цьому випадку легко довести такий досить загальний і сильний факт.

Теорема 4.6.3. Для всякого планарного графа G виконується нерівність $X_p(G) \leq 6$.

Доведення виконується індукцією за числом вершин графа G і майже дослівно повторює доведення теореми 4.6.2.

Детальніший аналіз шляхів зниження верхньої межі хроматичного числа приводить до так званої *теореми про п'ять фарб*.

Теорема 4.6.4. Для всякого планарного графа G справедливе нерівність $X_p(G) \leq 5$.

Доведення ведеться індукцією за числом n вершин графа G . Для планарних графів порядку менше ніж шість результат очевидний.

Нехай G — планарний граф з n вершинами, і для всіх планарних графів з $n - 1$ вершинами теорема справедлива. Можна вважати, що G — плоский граф і що він має згідно з наслідком 4.5.4 вершину v , степінь якої не більший 5. Як і раніше, вилучення вершини v з графа G приводить до графа з $n - 1$ вершинами, який згідно з індуктивним припущенням можна розфарбувати п'ятьма фарбами. Отже, наша задача звелася до того, щоб розфарбувати v одним із п'яти кольорів і на цьому закінчити доведення.

Якщо $n(v) < 5$, то v можна пофарбувати будь-якою фарбою, яка не бере участі у фарбуванні суміжних вершин.

Розглянемо випадок $n(v) = 5$. Нехай суміжні вершини v_1, v_2, \dots, v_5 розміщені навколо v за годинниковою стрілкою, як це показано на рис. 4.6.2. Якщо будь-яким двом вершинам v_i і v_j приписаний один і той же колір, то цим все доведено, оскільки v можна пофарбувава-

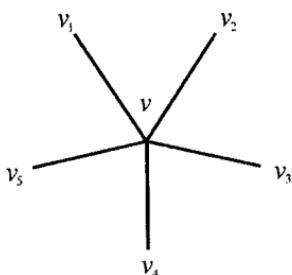


Рис. 4.6.2

ти кольором, що не використовувався для жодної з вершин v_i .

У цьому разі приходимо до останнього випадку, коли всім вершинам v_i приписані різні кольори. Нехай v_i пофарбована кольором c_i ($1 \leq i \leq 5$). Нехай граф H_{ij} — підграф граfa G , вершинами якого є всі вершини кольору c_i , а ребрами — всі ребра, що з'єднують вершину кольору c_i з вершиною кольору c_j . Тепер можуть бути лише два випадки.

1. Вершини v_1 і v_3 не належать одному і тому ж компоненту H_{13} (див. рис. 4.6.3).

У цьому випадку можна міняти колір всіх вершин компонента H_{13} , в який не входить вершина v_1 (колір c_1 міняється на c_3 і навпаки). В результаті вершина v_1 одержує колір c_3 , а це дозволяє вершину v пофарбувати кольором c_1 і закінчити доведення теореми для даного випадку.

2. Вершини v_1 і v_3 належать одному компоненту графа H_{13} . У цьому випадку існує цикл c , що має вигляд $v, v_1, v_2, \dots, v_3, v$. Частина його, обмежена вершинами v_1 і v_3 , повністю лежить всередині H_{13} (див. рис. 4.6.4).

Оскільки v_2 лежить всередині циклу c , а v_4 — зовні, то не існує простого ланцюга з v_2 в v_4 , який повністю належить H_{24} . Отже, можна поміняти кольори всіх вершин компонента підграфа H_{24} , якому належить вершина v_2 . При цьому вершина v_2 приймає колір v_4 , що дає можливість пофарбувати вершину v кольором c_2 .

Теорема доведена.

Логічно поставити питання: чи можна одержати сильніший результат, ніж результат, наведений в теоремі 4.6.4? Це питання приводить нас до однієї з найвідоміших нерозв'язаних проблем теорії графів — гіпотези чотирьох фарб.

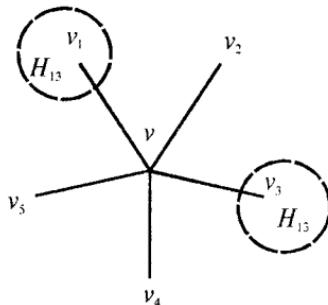


Рис. 4.6.3.

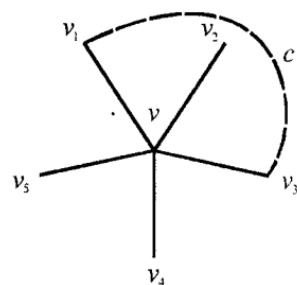


Рис. 4.6.4.

4. ГІПОТЕЗА ЧОТИРЬОХ ФАРБ

Гіпотеза чотирьох фарб формулюється так: всякий планарний граф розфарбовується чотирма фарбами.

Вже більше століття математики намагаються розв'язати дану

проблему (довести справедливість або хибність гіпотези), але досі остаточного розв'язку так і не знайдено. Відомо, наприклад, що всякий планарний граф порядку < 52 розфарбовується за допомогою чотирьох фарб.

Історично гіпотеза чотирьох фарб пов'язана з розфарбуванням географічних карт. Якщо у нас є географічна карта кількох країн світу, то цікаво знати, скільки фарб необхідно мати для такого розфарбування території цих країн, щоб жодна пара сусідніх країн не була пофарбована одним і тим же кольором. Можливо, найбільш звична форма гіпотези чотирьох фарб така: всяку карту можна розфарбувати чотирма фарбами.

Щоб зробити це твердження точним, необхідно дати точне формулювання поняття *карта*, оскільки в розглянутій нами задачі про розфарбування географічних карт потрібно, щоб країни, розміщені по обидві сторони ребра, були розфарбовані різними кольорами і не мали мостів. Таким чином, *карта* — зв'язний плоский граф, який не має мостів. Будемо говорити, що карта розфарбовується k кольорами, якщо її грані, що мають спільне ребро, можна розфарбувати k кольорами так, щоб ніякі дві суміжні грані не були одного кольору.

5. ГІПОТЕЗА ЧОТИРЬОХ ФАРБ ДЛЯ КАРТ

Гіпотеза чотирьох фарб для карт формулюється так: всяка карта розфарбовується чотирма фарбами.

Справедливе таке твердження, яке наводиться без доведення.

Теорема 4.6.5. *Гіпотеза чотирьох фарб для карт еквівалентна гіпотезі чотирьох фарб для планарних графів.*

Як видно з описаних вище результатів, задачі визначення хроматичного числа графа і побудови мінімального розфарбування довільного графа досить складні, а ефективні алгоритми їх розв'язку невідомі. Розглянемо простий алгоритм побудови правильного розфарбування, який у деяких випадках дає розфарбування, близькі до мінімальних.

РОЗФАРБ (V, E).

Початок.

1. Довільній вершині v_1 із V приписати колір 1.
2. Якщо вершини v_1, v_2, \dots, v_i розфарбовані r кольорами 1, 2, ..., r , де $r \leq i$, то новій довільно вибраній вершині v_{i+1} приписати мінімальний колір, який не використовується при розфарбуванні вершин із множини $C_m(v_{i+1})$.

Кінець.

На закінчення цього параграфа зауважимо, що розфарбування, яке виконує даний алгоритм, називається **послідовним**. Очевидно, що це розфарбування правильне. Для повних k -частинних графів послідовне розфарбування є мінімальним. У загальному випадку це не так.

§ 4.7. НЕСКІНЧЕННІ ГРАФИ

Для повноти картини розглянемо деякі поняття теорії нескінчених графів. Ці поняття, в основному, є узагальненнями понять, які були розглянуті вище для скінчених графів.

Нескінченим (неорієнтованим) графом називається пара $G = (V, E)$, де V — нескінчена множина вершин і E — нескінчена множина ребер. Якщо обидві множини V і E зліченні, то граф називається **зліченним**. Слід зазначити, що ці означення не стосуються того випадку, коли V — скінчена множина, а E — нескінчена множина, і коли E — скінчена множина, а V — нескінчена множина. У цих випадках граф G можна вважати скінченим графом, якщо він має в першому випадку скінченне число вершин і нескінченне число кратних ребер (або петель), а в другому випадку — скінченне число ребер і нескінченне число ізольованих вершин.

Означення таких понять, як суміжні вершини, інцидентні ребра, ізоморфні графи, підграф граfa, зв'язний граф, компонента зв'язності повністю переносяться на нескінченні графи.

Степенем вершини у нескінченноного графа називається потужність множини ребер, інцидентних вершині v . Отже, степінь вершини в нескінченному графі може бути як скінченим, так і нескінченим.

Нескінчений граф називається **локально скінченим**, якщо степінь кожної його вершини скінчений. Прикладом локально скінченноного графа може бути квадратна решітка, складена з цілих чисел (див. рис. 4.7.1).

Аналогічно дається означення локально зліченного нескінченноного графа: нескінчений граф називається **локально зліченним**, якщо кожна вершина цього графа має злічений степінь.

Теорема 4.7.1. *Будь-який зв'язний локально злічений нескінчений граф є зліченим.*

Доведення. Нехай v — довільна вершина такого нескінченноного графа і

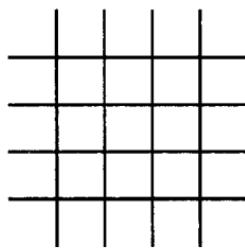


Рис. 4.7.1. Локально скінчений граф

нехай A — множина всіх вершин, суміжних з вершиною v ($A = \text{См}(v)$), B — множина всіх вершин, суміжних з вершинами з множини A і т.д. За умовою теореми множина A зліченна, множина B зліченна і т.д. За теоремою 1.1.10 об'єднання цих множин є множиною зліченою. Оскільки послідовність $\{v\}, A, B, \dots$ внаслідок зв'язності графа містить кожну вершину нескінченого графа, то цим теорема доведена повністю.

Наслідок 4.7.1. Всякий зв'язний локально скінчений нескінчений граф є зліченним.

Доведення випливає з того, що об'єднання зліченої множин скінчених множин є зліченою множиною.

На нескінченні графи переноситься і поняття маршруту, причому це поняття може мати такі різновиди:

1) *скінчений маршрут* у нескінченому графі G визначається так, як і у випадку скінченних графів;

2) *нескінченим в одну сторону маршрутом* у графі G , який починається у вершині v_0 , називається нескінчена послідовність ребер $(v_0, v_1), (v_1, v_2), (v_2, v_3), \dots$;

3) *нескінченим в обидві сторони маршрутом* в графі G називається нескінчена послідовність ребер ..., $(v_{-2}, v_{-1}), (v_{-1}, v_0), (v_0, v_1), (v_1, v_2), \dots$

Нескінченні в одну і обидві сторони ланцюги і прості ланцюги визначаються так, як і поняття довжини ланцюга і відстані між вершинами. Умови існування наскінчених ланцюгів у графах дає теорема, яка в теорії графів відома під назвою леми Кьоніга.

Лема Кьоніга. *Нехай G — зв'язний локально скінчений нескінчений граф. Тоді для всякої його вершини v існує нескінчений в однин бік простий ланцюг, який починається у вершині v .*

Д о в е д е н н я. Якщо v — довільна вершина графа G , відмінна від вершини v , то існує деякий простий ланцюг, який з'єднує вершини v і v . Звідси випливає, що в G повинно бути нескінчено багато простих ланцюгів з початком у вершині v . Оскільки степінь вершини v скінчений, то нескінчена множина таких простих шляхів повинна починатися з одного і того ж ребра. Якщо таким ребром є ребро (v, v_1) , то, повторивши цю процедуру для вершини v_1 , одержимо нову вершину v_2 і відповідне їй ребро (v_1, v_2) . Продовжуючи цей процес далі, одержимо нескінчений в одну сторону простий ланцюг v, v_1, v_2, \dots

Лема доведена.

Важливе значення леми Кьоніга полягає в тому, що вона дає можливість одержувати результати про нескінченні графи з відповідних результатів про скінченні графи.

Розглянемо на закінчення даного розділу нескінченні ейлерові та гамільтонові графи.

Зв'язний злічений граф G називається *ейлеровим*, якщо в ньому існує нескінчений в обидва боки ланцюг, який містить кожне ребро графа G . Такий ланцюг називається (двостороннім) *ейлеровим ланцюгом*.

Злічений граф G називається *напівейлеровим*, якщо в ньому існує нескінчений в одну або в обидва боки ланцюг, який містить в собі кожне ребро графа G .

Теорема 4.7.2. Нехай G — зв'язний злічений ейлерів граф; тоді:

а) в G кожна вершина має нескінчений або парний степінь;

б) для всякого скінченного підграфа H графа G доповнення \bar{H} графа H в графі G має не більше двох компонентів зв'язності;

в) якщо H — скінчений підграф графа G і кожна вершина підграфа H має парний степінь, то доповнення \bar{H} має рівно один нескінчений компонент зв'язності.

Д о в е д е н н я. Доведемо а). Нехай P — ейлерів ланцюг графа G . Тоді при будь-якому проходженні по ланцюгу P через довільну вершину графа G степінь цієї вершини збільшується на два. Оскільки кожне ребро зустрічається в P тільки один раз, то всяка вершина графа G має або нескінчений (злічений) степінь, або парний.

Доведемо б). Нехай P — ейлерів ланцюг графа G . Розіб'ємо P на три підланцюги: P_- , P_0 і P_+ так, щоб P_0 являв собою скінчений ланцюг, який містить всі ребра графа H (і, можливо, ще деякі ребра графа G , що не належать графу H), а P_- і P_+ — два нескінчених в один бік ланцюги. Тоді нескінчений граф K , вершинами якого є вершини P_+ і P_- , а ребрами — ребра цих ланцюгів, має не більше двох компонентів зв'язності. Оскільки H можна одержати з графа K введенням лише скінченного числа ребер, то звідси випливає, що і \bar{H} теж може мати не більше двох компонентів зв'язності.

При доведенні в) розіб'ємо, як і в попередньому випадку, ейлерів ланцюг P на три підланцюги: P_- , P_0 і P_+ . Нехай u — початкова, v — кінцева вершина ланцюга P_0 . Покажемо, що u і v з'єднані маршрутом в H .

Якщо $u = v$, то все ясно, а якщо $u \neq v$, то, вилучивши ребро (u, v) графа H , одержимо, що вершини u і v мають непарні степені в H . А звідси за наслідком 4.3.2 випливає, що u і v зв'язані напівейлеровим ланцюгом. Отже, \bar{H} має рівно один компонент зв'язності.

Теорема доведена.

Контрольні питання

1. Дайте означення:
 - а) ейлерового графа, ейлерового циклу;
 - б) гамільтонового графа, гамільтонового циклу;
 - в) хроматичного числа;
 - г) розрізу в графі, мосту.
2. Який граф називається: а) біхроматичним; б) k -хроматичним?
3. Які властивості мають зв'язні, ейлерові та гамільтонові графи?
4. Який граф називається планарним?
5. Сформулюйте критерій планарності графа.
6. Чи будь-який граф можна розфарбувати 4-, 5-, 6-ма фарбами?

Задачі і вправи

1. Нехай V — множина додатних цілих чисел від 1 до 20 і $a|b$ — відношення подільності, тобто $a|b$ означає, що число a є дільником числа b . Намалюйте граф відношення $a|b$ для множини V .

2. Побудуйте матриці суміжності та інцидентності для платонових графів.

3. Які з платонових графів мають ейлерові цикли, гамільтонові цикли?

4. Знайдіть ейлерів ланцюг для графа, зображеного на рис. 4.3.1.

5. Знайдіть хроматичні числа:

а) платонових графів;

б) об'єднання двох графів з хроматичними числами k і k' відповідно;

в) з'єднання двох графів з хроматичними числами k і k' відповідно.

6. Нехай G — регулярний граф порядку n і степеня d . Покажіть, що $X_p(G) \geq n/(n-d)$.

7. Граф називається **критичним**, якщо вилучення будь-якої його вершини приводить до графа з меншим хроматичним числом.

Доведіть, що:

а) K_n є критичним для всіх $n > 1$;

б) C_n критичний тоді і тільки тоді, коли n непарне;

в) якщо n непарне, то з'єднання C_n і C_n є критичним графом, хроматичне число якого дорівнює 6.

8. Покажіть, що будь-який k -хроматичний критичний граф $G = (V, E)$ має такі властивості:

а) G — зв'язний;

б) $(\forall v \in V) n(v) \geq k - 1$;

в) не існує вершини v із V , вилучення якої веде до незв'язного графа.

9. Перевірте теорему Ейлера для:

а) платонових графів; б) W_n ; в) $K_{2,n}$;

г) графа, який являє собою шахова дошка.

§ 4.8. ДЕРЕВА

Перейдемо до розгляду одного з найпоширеніших різновидів графів — дерев.

1. ОЗНАЧЕННЯ ДЕРЕВА. ВЛАСТИВОСТІ ДЕРЕВ

Зв'язний ацикличний граф називається (неоріентованим) **деревом**. Древо називається **кореневим**, якщо в ньому виділена вершина, яка називається **коренем**. **Остовним деревом** графа G називається остатовний підграф графа G , який є деревом.

Безпосередньо з означення дерева випливає така теорема.

Теорема 4.8.1. *Граф є деревом тоді і тільки тоді, коли будь-які дві його вершини зв'язані лише одним ланцюгом.*

Д о в е д е н н я. Нехай граф є дерево. Якщо допустити існування більше одного ланцюга, який з'єднує довільні дві його вершини, то в такому графі існує цикл, тобто граф не може бути деревом.

Навпаки, оскільки довільні дві вершини графа з'єднані ланцюгом, то граф зв'язний, а через те, що цей ланцюг єдиний, він не має циклів. Значить, граф є деревом.

Теорема доведена.

Наслідок 4.8.1. Якщо T — дерево і u — його кінцева вершина, то граф $T - u$ — дерево.

Дійсно, граф $T - u$ — підграф дерева T , для якого виконуються всі умови теореми 4.8.1.

Наслідок 4.8.2. Всяке непусте дерево має щонайменше дві кінцеві вершини і одне кінцеве ребро.

Доведення виконується індукцією за числом вершин дерева з наступним застосуванням леми про рукостискання.

Ребро зв'язного графа називається **суттєвим**, якщо його вилучення веде до порушення зв'язності цього графа.

Наслідок 4.8.3. В дереві кожне ребро суттєве.

Доведення випливає з того, що коли вилучити ребро (u, v) в дереві T , то оскільки вершини u і v з'єднує лише один ланцюг, будемо мати два компоненти зв'язності: один містить вершину u , другий — вершину v . Отже, граф T не є зв'язним.

Теорема 4.8.2. Якщо $T = (V, E)$ — дерево і вершина $v \notin V$, то граф $T' = (V \cup \{v\}, E \cup \{(u, v)\})$, де u — довільна вершина із V , теж є деревом.

Д о в е д е н н я. Оскільки T — дерево, то існує єдиний ланцюг, який з'єднує довільну вершину u' з вершиною u . Оскільки вершина $v \notin V$, то введення одного кінцевого ребра (u, v) приводить до того, що з кожної вершини u' маємо лише один ланцюг,

який з'єднує вершини u і v . За теоремою 4.8.1 граф T' є деревом.

Перелічені вище властивості дерев підсумовує така теорема

Теорема 4.8.3. *Нехай граф T має n вершин. Тоді еквівалентними є такі твердження:*

- (1) T є деревом;
- (2) T не має циклів і має $n - 1$ ребро;
- (3) T — зв'язний граф і має $n - 1$ ребро;
- (4) T — зв'язний граф, і кожне його ребро є мостом;
- (5) будь-які дві вершини графа T з'єднані рівно одним простим ланцюгом;
- (6) T не має циклів, але введення нового ребра в T сприяє виникненню рівно одного циклу.

Д о в е д е н н я. Випадок, коли $n = 1$, очевидний. Нехай $n \geq 2$.

(1) \rightarrow (2). *Індукція за числом n .* Для $n = 2$ твердження очевидне. Оскільки T не має циклів, то вилучення будь-якого ребра з T (згідно з наслідком 4.8.3) веде до розбиття T на два підграфи, кожний з яких — дерево. За індуктивним припущенням у кожному з цих підграфів ребер на одиницю менше, ніж вершин. Отже, число всіх ребер в T дорівнює $n_1 - 1 + n_2 - 1 + 1 = n - 1$.

(2) \rightarrow (3). Якщо T незв'язний, то кожен його компонент зв'язності T_i являє собою зв'язний граф без циклів. З (2) випливає, що кожен компонент зв'язності T_i має $n_i - 1$ ребро. Але тоді граф T має не більше $n - 2$ ребра, що суперечить умові. Отже, граф T зв'язний і має $n - 1$ ребро.

(3) \rightarrow (4). Вилучення будь-якого ребра із T веде до графа, який має n вершин і $n - 2$ ребра. Але такий граф за наслідком 4.8.3 не може бути зв'язним.

(4) \rightarrow (5). Оскільки T зв'язний, то всяка пара його вершин з'єднана хоча б одним простим ланцюгом. Якщо ж деяка пара вершин з'єднується двома простими ланцюгами, то вони складають цикл, а це суперечить тому, що кожне ребро в T є мостом.

(5) \rightarrow (6). Якщо T має цикл, то будь-які дві вершини цього циклу з'єднані не менше, ніж двома простими ланцюгами. Добаючи до графа T деяке ребро e , одержуємо цикл, оскільки вершини, інцидентні ребру e , уже з'єднані в T простим ланцюгом. Покажемо, що цей цикл єдиний. Дійсно, нехай вершини u і v несуміжні. За умовою u і v з'єднані єдиним простим ланцюгом. Якщо ввести ребро (u, v) в T , то простий ланцюг переходить у простий цикл. Якби існував при цьому ще деякий цикл, який містить ребро (u, v) , то це суперечило б тому, що u і v з'єднані єдиним простим ланцюгом. Якщо u і v суміжні, то теорема, очевидно, справедлива (з'являються кратні ребра, а це цикл).

(6) \rightarrow (1). Нехай T — незв'язний граф. Тоді введення довільного ребра, яке з'єднує вершини з різних компонентів зв'язності, не приводить до появи циклу, а це суперечить умові (6). Отже, граф T не має циклів і зв'язний, тобто T — дерево.

Теорема доведена.

Наслідок 4.8.4. Нехай G — ліс з n вершинами і k компонентами; тоді G має $n - k$ ребер.

Д о в е д е н н я. З умови (2) теореми 4.8.3 випливає, що кожен компонент G_i має $n_i - 1$ ребро. Але тоді число ребер в G становить

$$(n_1 - 1) + (n_2 - 1) + \dots + (n_k - 1) = n_1 + n_2 + \dots + n_k - k = n - k,$$

що й потрібно було показати.

Часто виникає питання: скільки можна побудувати різних дерев порядку n . Відповідь на це питання дає теорема Келі.

Теорема Келі. Число різних дерев, які можна побудувати на n вершинах, дорівнює n^{n-2} .

Д о в е д е н н я. Нехай $V = \{v_1, v_2, \dots, v_n\}$ і $T = (V, E)$ — дерево. За наслідком 4.8.2 в T повинні бути кінцеві вершини. Нехай v_1 — перша з них і (u_1, v_1) — відповідне їй кінцеве ребро. Вилучимо з T це ребро і вершину v_1 , позначимо в множині V вершину u_1 і занесемо її в список вершин L , який спочатку є пустим. За наслідком 4.8.1 одержаний граф знову є деревом. Застосуємо до нього цю ж саму процедуру, одержимо нове дерево і список $L = [u_1, u_2]$ і т. д. Цю процедуру застосовуємо доти, доки після вилучення ребра (u_{n-2}, v_{n-2}) не залишиться єдине ребро (u_{n-1}, v_{n-1}) . Тоді список $L = [u_1, u_2, \dots, u_{n-2}]$ однозначно визначається деревом T , і двом різним деревам T і T' з n вершинами відповідають різні списки. Кожна вершина u з'являється в списку L $n(u) - 1$ раз.

Навпаки, кожний список L визначає дерево T за допомогою зворотної побудови. Якщо задано список L , то знаходимо першу позначену вершину v_1 в множині V , таку, що $v_1 \in L$. Ця вершина визначає ребро (u_1, v_1) . Далі витираємо позначку вершини u_1 в множині V , вилучаємо u_1 із L і продовжуємо побудову для нового списку L , який складається з $n - 3$ елементів. Одержаній внаслідок такої побудови граф є деревом згідно з теоремою 4.8.2.

Оскільки різних елементів у списку L може бути не більше n^{n-2} , то цим і доводиться справедливість теореми.

Теорема Келі має багато різних інтерпретацій. Зупинимось на одній з них, яка відома під назвою *задачі про з'єднання міст*: необхідно з'єднати n міст залізницями так, щоб не будувати зайвих ліній; скількома способами можна побудувати таку систему дір?

Практичне значення цієї задачі полягає у тому, що: нехай відома вартість $c(a, b)$ будівництва залізниці між містами a і b ; яка мережа доріг, що з'єднує всі n міст, має найменшу можливу вартість?

Теорема Келі дає розв'язок цієї задачі. Для цього кожному дереву T , побудованому на n вершинах, ставиться у відповідність сума

$$C(T) = \sum_{(a, b) \in E} c(a, b).$$

Вибравши серед цих сум найменшу, одержимо розв'язок задачі. Для цього, як випливає з теореми Келі, необхідно переглянути не більше n^{n-2} сум.

Як відомо з попереднього, вилучення в зв'язному графі G одного ребра, що належить деякому циклу, не порушує зв'язності графа G (див. теорему 4.3.8). Застосуємо цю процедуру до одного з циклів, які залишилися в графі G , і так до тих пір, поки в G не залишиться жодного циклу. Одержано дерево, яке містить всі вершини графа G . Це дерево називається *остовним деревом графа G* .

Нехай G являє собою граф з n вершинами, m ребрами і k компонентами. Застосовуючи наведену вище процедуру до кожного компонента графа G , одержимо граф, який називається *остовним лісом*. Число ребер, які при цьому вилучаються, називається *цикломатичним числом* або *циклічним рангом графа G* і позначається $C(G)$. Таким чином, цикломатичне число є мірою зв'язності графа.

Слід зауважити, що циклічний ранг дерева дорівнює нулю, а циклічний ранг циклічного графа — одиниці.

Нехай T — оставний ліс графа G . Граф G' , одержаний із графа G шляхом вилучення всіх ребер графа T , називається *доповненням оставного лісу T графа G* .

Теорема 4.8.4. Якщо T — оставний ліс графа G , то:

- (а) *всякий розріз в G має спільне ребро з T ;*
- (б) *всякий цикл в G має спільне ребро з доповненням T .*

Д о в е д е н н я. Доведемо (а). Нехай M — розріз графа G , вилучення якого розбиває один із компонентів G на два підграфи: H і F . Оскільки T — оставний ліс, то в ньому повинно існувати ребро, яке з'єднує вершину з графа H з вершиною з графа F . Але це ребро є шуканим ребром.

Доведемо (б). Якщо C — цикл у графі G , який не має жодного спільногого ребра з доповненням T , то C входить до оставного лісу, а це неможливо, оскільки T — ліс.

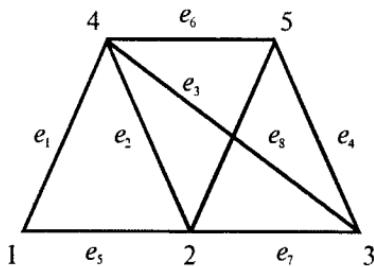
Теорема доведена.

2. ФУНДАМЕНТАЛЬНА СИСТЕМА ЦІКЛІВ ГРАФА

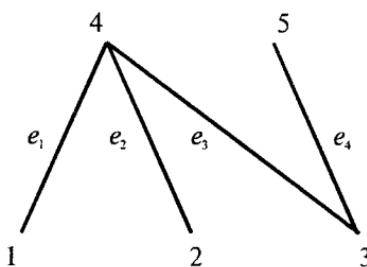
З поняттям остворного лісу T графа G тісно пов'язане поняття фундаментальної системи циклів, яка асоціюється з T . Нехай T — остворний ліс графа G . Якщо ввести довільне ребро в граф G , яке не входить в T , то згідно з п. (6) теореми 4.8.3 одержимо єдиний цикл. Множина всіх циклів, які одержані в такий спосіб (шляхом введення окремо кожного ребра в граф G , яке не входить до T), називається *фундаментальною системою циклів*, асоційованою з T . У тому випадку, коли нас не цікавить, який остворний ліс розглядається, будемо говорити про фундаментальну систему циклів графа G .

Приклад 4.8.1

Система графів на рис. 4.8.1 демонструє введене поняття.



Граф G



Остворне дерево графа G

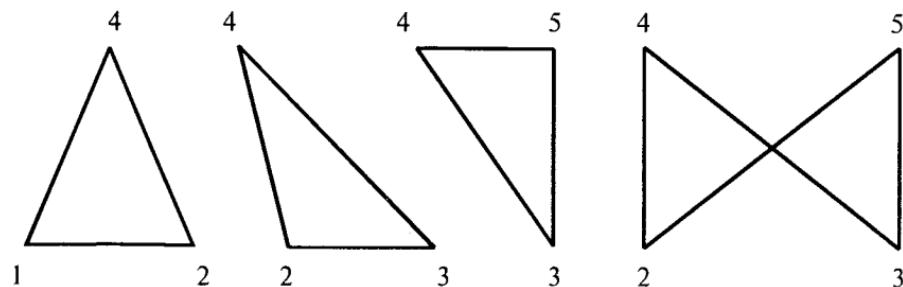


Рис. 4.8.1.

Фундаментальна система циклів, асоційована
з остворним деревом графа G . ▲

Теорема 4.8.5. Нехай $G = (V, E)$ — довільний скінченний граф. Число ребер графа G , які необхідно вилучити для одержання оствор-

ногого лісу T , не залежить від порядку їх вилучення і становить $C(G) = |E| - |V| + k$, де k — число компонент зв'язності графа G .

Доведення. Нехай $G_i = (V_i, E_i)$ ($i = 1, 2, \dots, k$) — один з компонентів зв'язності графа G . З теореми 4.8.3 (п. (2)) випливає, що для одержання оствового дерева T_i графа G_i необхідно вилучити $|E_i| - (|V_i| - 1)$ ребро. Тоді загальна кількість ребер, які необхідно вилучити в графі G , становить

$$(|E_1| - (|V_1| - 1)) + (|E_2| - (|V_2| - 1)) + \dots + (|E_k| - (|V_k| - 1)) = |E| - |V| + k.$$

Теорема доведена.

Наслідок 4.8.5. Граф G являє собою ліс тоді і тільки тоді, коли $C(G) = 0$.

Наслідок 4.8.6. Граф G має єдиний цикл тоді і тільки тоді, коли $C(G) = 1$.

Наслідок 4.8.7. Граф G , в якого число ребер перевищує число вершин, має цикл.

Наслідок 4.8.8. Всяке дерево порядку $n \geq 2$ має щонайменше дві кінцеві вершини.

Доведення. З леми про рукостискання і теореми 4.8.3 (п. (2)) випливає, що сума всіх степенів вершин дерева дорівнює $2(n - 1)$ і кожен степінь вершини більший від нуля. Значить, хоча б два числа із всіх степенів вершин дерева дорівнюють одиниці.

Теорема 4.8.6. Всякий ациклічний підграф довільного скінченного графа $G = (V, E)$ є підграфом оствового дерева графа G .

Доведення. Нехай $H = (V', E')$ — ациклічний підграф графа G . Достатньо розглянути ситуацію, коли G зв'язний. Припустимо, що H незв'язний і H' — одна з його компонент зв'язності. Тоді в G існує таке ребро (u, v) , що $u \in A$, а $v \in V' \setminus A$, де A — множина вершин графа H' . У такому разі граф $H + (u, v)$ — ациклічний підграф графа G , який має менше компонент зв'язності, ніж H . Повторюючи цю побудову далі, ми врешті-решт прийдемо до оствового дерева графа G , яке містить H як підграф.

Нехай тепер H зв'язний, але він не є оствовим деревом графа G . Оскільки H — зв'язний ациклічний граф, то H — дерево, і оскільки H не є оствовим деревом графа G , то в G існує вершина V , яка не належить H , і в G існує ребро (u, v) , яке з'єднує вершину v з однією з вершин графа H . Тоді граф $H + (u, v)$ являє собою дерево, в якому H є підграфом. Якщо $H + (u, v)$ — оствовне дерево G , то все доведено, а якщо ні, то, повторюючи дану побудову, ми, нарешті, прийдемо до оствового дерева графа G , для якого H буде підграфом.

І якщо H — оствовне дерево для G , то і в цьому випадку теорема справедлива.

Теорема доведена.

3. ОСТОВ НАЙМЕНШОЇ ВАГИ

Розглянемо таку задачу: нехай кожному ребру графа $G = (V, E)$ поставлено у відповідність деяке число $d_i = d(e_i)$, $e_i \in E$, $i = 1, 2, \dots, |E|$; необхідно в графі G знайти оствовне дерево, сума чисел d_i в якому найменша. Число d_i в цьому випадку називається **вагою ребра** e_i , а сам граф G — таким, що має **вагу**. Отже, задача полягає в тому, щоб знайти оствовне дерево найменшої ваги. Ця задача виникає при проектуванні доріг, ліній електропередач, трубопроводів і т. ін., коли необхідно з'єднати задані точки деякою системою зв'язку так, щоб будь-які дві точки були зв'язані безпосередньо між собою, або через інші точки, і щоб загальна довжина ліній зв'язку була мінімальною. Розглянемо кілька методів розв'язання цієї задачі.

Перший розв'язок випливає безпосередньо з теореми Келі. Дійсно, розглянувши всі можливі оствовні дерева повного графа G з n вершинами (а їх буде не більше, ніж n^{n-2}) і вибравши те з них, яке має мінімальну суму ваг, розв'яжемо дану задачу. Але оскільки число n^{n-2} навіть при невеликому n буде великим, то такий шлях розв'язання даної задачі досить громіздкий. Розглянемо ефективніші способи розв'язання задачі. Одним із них є спосіб, або алгоритм, Краскала.

Алгоритм Краскала полягає у виконанні такої послідовності дій:

1) беремо пустий граф O і будуємо граф $T_1 = O + e_1$, де e_1 — ребро графа $G = (V, E)$ мінімальної ваги;

2) якщо граф T_k уже побудований і $k < n - 1$, то будуємо граф $T_{k+1} = T_k + e_{k+1}$, де e_{k+1} — ребро мінімальної ваги серед тих ребер графа G , які не ввійшли в граф T_k , і яке не утворює цикл з ребрами графа T_k .

Цей алгоритм базується на такій теоремі.

Теорема 4.8.7. *Нехай G — зв'язний граф порядку n і T — підграф графа G , одержаний внаслідок виконання кроків 1) і 2) алгоритму Краскала. Тоді підграф T — оствовне дерево графа G мінімальної ваги.*

Д о в е д е н н я. Те, що T — оствовне дерево графа G , випливає з теореми 4.8.3 (п. (2)). Залишається лише показати, що сума чисел d_i для всіх ребер T мінімальна. Припустимо супротивне, тобто що існує дерево S графа G , таке, що його вага $P(S)$

менша ваги дерева $T = P(T)$. Якщо e_i — перше ребро, яке не належить графу S , то підграф граfa G , який дорівнює $S + e_i$, матиме єдиний цикл C , який включає ребро e_i . Оскільки C включає ребро e , яке належить S і не належить T , то підграф S' , одержаний із S шляхом заміни e на e_i , також буде оствовним деревом S граfa G . За побудовою $d(e_i) \leq d(e)$ і тому $P(S') \leq P(S)$, причому число спільних ребер у S' і T на одне більше, ніж у S і T . Повторюючи крок за кроком цю процедуру, можна перетворити S в T так, щоб сума чисел d_i на кожному кроці не збільшувалась. Отже, $P(T) \leq P(S)$ і ми одержали суперечність.

Теорема доведена.

Приклад 4.8.2

Розглянемо граf, зображенний на рис. 4.8.2, і знайдемо оствовне дерево цього граfa.

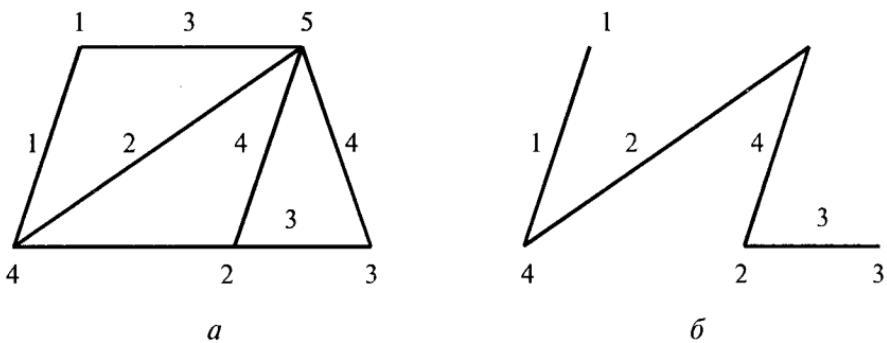


Рис 4.8.2. Граf G (a) і
остовне дерево граfa G мінімальної ваги (b).

§ 4.9. ОРІЄНТОВАНІ ГРАФИ І ДЕРЕВА

1. ОСНОВНІ ПОНЯТТЯ

Нагадаємо, що коли ребра граfa $G = (V, E)$ задані у вигляді упорядкованих пар (u, v) , тобто $E \subseteq V^2$, то граf G називається **орієнтованим** (орграфом). Ребра орграфа називаються **дугами**, вершина u — **початком**, вершина v — **кінцем дуги**. Дуга (u, v) називається дугою, що виходить із вершини u і веде у вершину v .

Нехай $G = (V, E)$ — орграф. Якщо $(u, v) \in E$, то вершина u називається **безпосереднім попередником** (предком) вершини v , а вершина v — **безпосередньо підлеглою вершині** і або **безпосередньо досяжною з вершини u** .

Число дуг $on(u)$, які виходять з вершини u , називається *степенем виходу вершини u* , а число дуг $in(u)$, які ведуть у вершину u , — *степенем входу вершини u* . Очевидно, що степінь вершини $n(u) = on(u) + in(u)$. Якщо $(\forall u \in V) on(u) \leq 2$, то граф G називається *бінарним*. Цікаво, як формулюється лема про рукостискання для орграфів. Оскільки кожна дуга орграфа з E “бере” участь у кожній із сум

$$\sum_{v \in V} in(v) \quad i \quad \sum_{v \in V} on(v)$$

лише по одному разу, то маємо таке співвідношення:

$$\sum_{v \in V} in(v) = \sum_{v \in V} on(v) = |E|.$$

Отже, лема про рукостискання для орграфів формулюється так.

Лема про рукостискання для орграфів. Сума степенів входу всіх вершин орграфа дорівнює сумі степенів виходу всіх його вершин.

Цей результат часто називають *орлемою про рукостискання*.

Маршрутом, або *шляхом* l із вершини u у вершину v ($l = l(u, v)$) в орграфі G називається послідовність вершин $u = u_1, u_2, \dots, u_n = v$, таких, що $(u_i, u_{i+1}) \in E$, $i = 1, 2, \dots, k - 1$. При цьому вершину u називають *початковою*, а v — *кінцевою вершиною* шляху l і говорять, що u досяжна з v по шляху $l(u, v)$ або що шлях l веде з вершини u у вершину v . Дуги, якими з'єднані вершини маршруту, називаються *ребрами маршруту*. Маршрут називається *орланцюгом*, якщо всі ребра цього маршруту різні, і *простим орланцюгом*, якщо всі вершини маршруту, крім першої і останньої вершин, різні. Орланцюг називається *замкнутим*, якщо перша і остання його вершини збігаються. *Цикл* — це простий замкнутий орланцюг.

Якщо u — вершина графа G , для якої $on(u) = 0$ ($in(u) = 0$), то така вершина називається, як і раніше, *кінцевою*, або *висячою вершиною*, або *стоком (джерелом)* графа G .

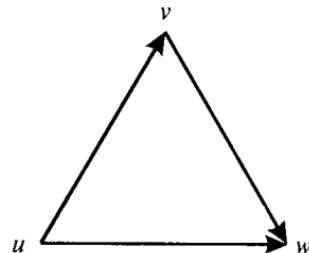


Рис. 4.9.1

Приклад 4.9.1

В графі на рис. 4.9.1 вершина u є джерелом, а вершина w — стоком.

Нехай R — відношення безпосередньої досяжності на множині V , а R — його транзитивне замикання. Очевидно, що відношення досяжності збігається з відношенням R . Орграф G називається **сильно зв'язним**, якщо $(\forall u, v \in V) uR_v$. Зв'язний орграф називається **ейлеровим орграфом**, якщо в ньому існує замкнений орланцюг, який проходить через кожну дугу графа.

Теорема 4.9.1. *Зв'язний орграф є ейлеровим тоді і тільки тоді, коли $in(v) = out(v)$ для будь-якої його вершини v .*

Доведення аналогічне доведенню теореми 4.3.10.

Орграф G називається **гамільтоновим**, якщо в ньому існує цикл, що проходить через кожну з його вершин. Орграф називається **напівгамільтоновим**, якщо в ньому існує простий орланцюг, який проходить через кожну з його вершин.

Для гамільтонових орграфів справа дещо ускладнюється, що має місце, як ми знаємо, і для неоріентованих графів. Логічно поставити питання: а чи узагальнюється теорема Дірака на гамільтонові орграфи? Одне таке узагальнення належить Гуйя-Урі.

Теорема 4.9.2. *Нехай G — сильно зв'язний орграф порядку n . Якщо $out(v) \geq n/2$ і $in(v) \geq n/2$ для будь-якої його вершини v , то G є гамільтоновим графом.*

Доведення теореми досить складне і виходить за рамки даної книги.

Ациклічним орграфом називається орграф, в якого немає циклів.

Орієнтованим деревом називається ациклічний орграф, який задовольняє такі умови:

- 1) $(\exists v_0 \in V) in(v_0) = 0$; вершина v_0 називається **коренем**;
- 2) $(\forall v \neq v_0) in(v) = 1$, де $v \in V$;
- 3) $(\forall v \neq v_0)$ існує шлях $v_0, v_1, \dots, v_n = v$, причому єдиний.

Нижче, коли не обумовлено супротивне, під деревом будемо розуміти орієнтоване дерево.

Орієнтований ліс — це орграф, який складається з кількох дерев. Якщо (u, v) — ребро дерева T , то вершина u називається **батьком вершини v** , а v — **сином вершини u** . Якщо $out(u) = 0$, то u називається **листком у дереві**. Вершина u разом зі всіма своїми нащадками становить піддерево дерева T . При цьому u — корінь цього піддерева.

Теорема 4.9.3. *Орграф $G = (V, E)$ є орієнтованим деревом, якщо:*

(а) *орграф G — зв'язний і існує $v_0 \in V$, яка не має предків, і для довільної вершини $u \in V$ $in(u) = 1$;*

(б) *орграф G має вершину v_0 , з якої досяжна будь-яка інша вершина u , причому шлях, який з'єднує v_0 і u , єдиний;*

(в) орграф G має вершину v_0 , яка не має предків, а решта вершин мають тільки одного безпосереднього предка, і з v_0 досяжна будь-яка інша вершина.

Доведення пропонується читачеві як проста вправа.

Упорядкованим деревом будемо називати дерево, в якого множина синів кожної вершини упорядкована.

Бінарне дерево T називається **повним**, якщо для кожної вершини u із V глибини, меншої k , $on(u) = 2^k$ і глибини k $on(u) = 0$, де **глибина вершини** u — це довжина шляху з кореня в цю вершину.

Висота вершини в дереві T — це довжина найдовшого шляху із заданої вершини у вершину-листок цього дерева, досяжний з даної вершини. **Висотою дерева** T називається висота його початкової вершини.

Рівень вершини — це різниця висоти дерева і глибини цієї вершини.

2. БІНАРНІ ВІДНОШЕННЯ І ОРГРАФИ

З означення орграфа випливає, що множину ребер E можна розглядати як бінарне відношення на множині V . Отже, всяке бінарне відношення R на множині V може розглядатись як орграф $G(V, R)$, якщо за множину ребер графа взяти множину R . Очевидно, що справедливе і обернене твердження: всякий орграф $G = (V, E)$ задає деяке бінарне відношення R на множині V . Дійсно, якщо визначити бінарне відношення так, що два елементи u і v із множини V знаходяться у відношенні R тоді і тільки тоді, коли $(u, v) \in E$, то відношення R і буде шуканим.

Таким чином, між бінарними відношеннями, визначеними на деякій скінченній множині V , і орграфами, для яких множина V є множиною вершин, існує взаємно однозначна відповідність.

Розглянемо орграфи, які відповідають відношенню еквівалентності.

Нехай R — відношення еквівалентності на V . Із властивостей відношення R випливає, що V розбивається на класи еквівалентності V_1, V_2, \dots, V_k , і внаслідок рефлексивності, симетричності і транзитивності відношення R маємо:

- 1) vRv , тобто $(v, v) \in E$;
- 2) $vRv_1 \Rightarrow (v_1Rv)$, тобто $(v, v_1) \in E$ і $(v_1, v) \in E$;
- 3) vRv_1 і $v_1Ru \Rightarrow (v, v_1) \in E$ і $(v_1, u) \in E \Rightarrow (v, u) \in E$,
 $(v, v_1), (v_1, u) \Rightarrow (v, u) \in E$.

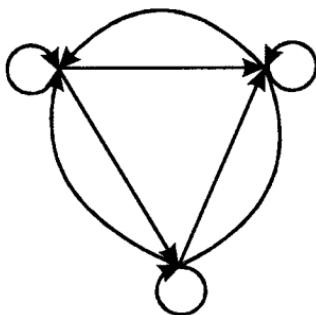


Рис. 4.9.2

Інакше кажучи, будь-який елемент у класі еквівалентності повинен бути зв'язаний із кожним елементом цього класу ребром і навпаки. Наприклад, якщо деякий клас має три вершини: v , u , w , то орграф цього класу еквівалентності матиме вигляд, показаний на рис. 4.9.2.

Підводячи підсумок, зауважимо, що відношення еквівалентності R на множині V породжує k орграфів $G_1 = (V_1, E_1), \dots, G_k = (V_k, E_k)$ (для кожного класу свій орграф).

3. ПОЗНАЧЕНИ ГРАФИ І ЗОБРАЖЕННЯ ТЕРМІВ

Нехай MV і ME — деякі множини, які будемо називати **множинами позначок**.

Граф $G = (V, E)$ називається **позначенням за допомогою функцій** f і g , якщо

$$\begin{aligned} f: V &\rightarrow MV; \\ g: E &\rightarrow ME. \end{aligned}$$

При цьому функція f називається **функцією позначення вершин**, а функція g — **функцією позначення ребер**.

Нехай $G = (V, E)$ і $H = (V', E')$ — позначені графи за допомогою функцій f , g і f' , g' відповідно.

Графи G і H називаються **еквівалентними**, якщо вони ізоморфні і коли h — цей ізоморфізм, то

$$\begin{aligned} f(u) &= f'(h(u)), \\ g((u, v)) &= g'((h(u), h(v))), \end{aligned}$$

тобто відповідні вершини і ребра мають одні і ті ж позначки.

Якщо функція f (g) тотожна, тобто $MV = V$ і $f(u) = u$ ($ME = E$ і $g((u, v)) = (u, v)$), то в графі позначені тільки ребра (тільки вершини). У цьому випадку в практичних застосуваннях функція f (g) просто не береться до уваги.

Розглянемо тепер конструкцію позначеного графа, яка зручна для задання термів.

Нехай $G = (V, E)$ — орграф, $G' = (A, \Omega)$ — деяка алгебра і \mathbf{N} — множина натуральних чисел. Покладемо $MV = \Omega$.

Складеним об'єктом називається четвірка $S = (V, L, f, v_0)$, де V — множина вершин, $v_0 \in V$ — початкова вершина, $f: V \rightarrow \Omega$ —

функція позначення, $L = V \times V \times N$ — множина упорядкованих зв'язків якщо $f(v) = \omega^n \in \Omega$, то $|L \cap (v, V, N)| \leq n$,

$$|L \cap (v, V, i)| \leq \begin{cases} 1, & i \leq n; \\ 0, & i > n. \end{cases}$$

Нехай R_i — транзитивне замикання відношення безпосередньої досяжності. Складений об'єкт називається ініціальним, якщо для будь-якої вершини $v \neq v_0 \Rightarrow v_0 R_i v$. Нижче будуть розглядається лише ініціальні складені об'єкти.

Умовимося надалі називати складений об'єкт просто складеним.

Граф $G = (V, E)$ буде називатися *графом складеного S*. Складений називається *деревом*, якщо його граф є деревом. Якщо $f(v) = \omega$, де ω — нульварна операція з Ω , то ω називають *первинним складеним*. Кожна вершина складеного S породжує складений, початковою вершиною якого є вона сама. Складений S_v , який породжується вершиною v' , називається *i-м аргументом складеного S*, ($\arg(v, i) = v'$), що породжений вершиною v , якщо існує $i \in N$ таке, що $(v, v', i) \in L$. Скориставшись відношенням безпосередньої досяжності, операцію \arg можна узагальнити на всю множину V , якщо покласти для $v_1, v_2, \dots, v_n \in V$ таких, що $\arg(v_k, i_{k+1}) = v_{k+1}$, справедлива рівність $\arg(v, i_1, \dots, i_k) = v_{k+1}$ ($k = 1, n - 1$). Позначкою складеного називається позначка його початкової вершини.

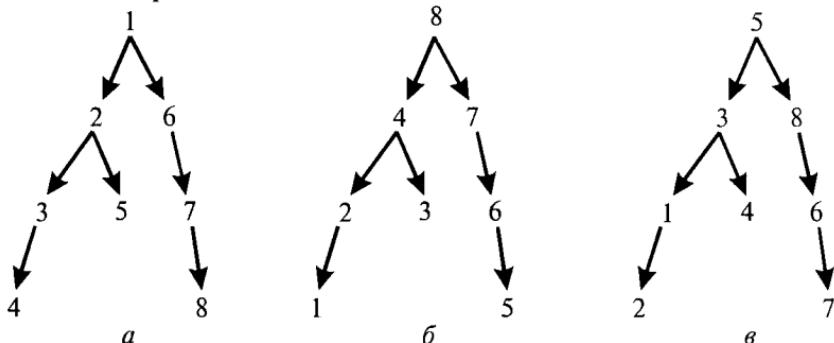


Рис. 4.9.3

Велика кількість задач як для графів, так і для дерев вимають обходу вершин графа (дерева) в деякому порядку. Найбільш поширені три способи обходу вершин графів: прямий, зворотний, внутрішній (рис. 4.9.3, відповідно a , b , c).

Частіше за інші використовується прямий порядок обходу вершин складеного, який ще називають *обходом зліва направо і зверху вниз*. Його суть полягає у виконанні таких дій:

- 1) відвідати початкову вершину складеного — v_0 ;
- 2) відвідати самий лівий аргумент v' вершини v , який ще не відвідувався $((v, v', i) \in L)$.

Відповідно до цього порядку нумеруються і вершини складеного. При цьому вважається, що номер вершини v ототожнюється з самою вершиною. Завдяки такому ототожненню складений об'єкт можна задавати за допомогою масиву (вершин) в пам'яті обчислювальної машини.

Складені об'єкти являють собою зручний спосіб для задання термів.

Приклади 4.9.2

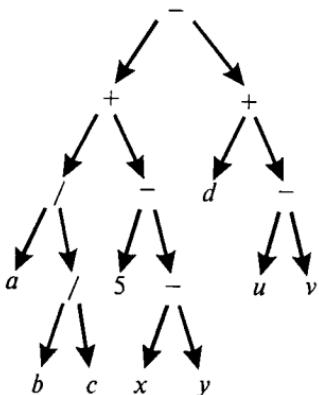


Рис. 4.9.4. Складений об'єкт терма t

1. Терм $t = (((a/(b/c)) + (5 - (x - y))) - (d + (u - v)))$ задається складеним об'єктом, зображенім на рис. 4.9.4.

2. Терм $t = (((x \cdot y) + x) + x \cdot y) + x \cdot y$ задається двома складеними, зображеніми на рис. 4.9.5 та 4.9.6.

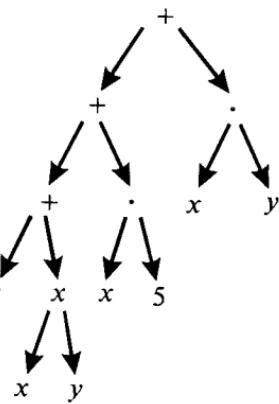


Рис. 4.9.5.
Складений, що представляє терм t , граф якого — дерево

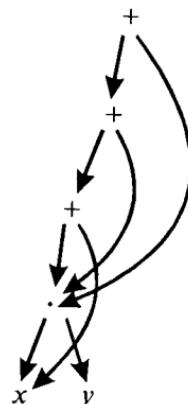


Рис. 4.9.6. Ациклічний складений, що представляє терм t

Зауважимо, що другий спосіб зображення терма набагато економніший, оскільки граф його складеного має лише 6 вершин і 8 дуг, в той час як перше зображення має на 7 вершин і на 4 дуги більше.

Контрольні питання

- Що таке граф, мультиграф, псевдограф?
- Що таке платонів граф, регулярний граф, повний граф, пустий граф, зв'язний граф, орієнтований граф?
- Що таке двочастинний граф, повний двочастинний граф, k -частинний граф?
- Яка різниця між графом і орграфом, графом і мультиграфом, графом і псевдографом?
- Чи буде двочастинним графом дерево, ациклічний граф (орграф)?
- Що означає запис $X_p(G) = 5$?
- Чи всякий граф можна розфарбувати 5-а різними фарбами?
- Скільки двочастинних графів можна побудувати на множині, яка складається з чотирьох елементів?
- Чи існує регулярний граф степеня 3 з трьома вершинами?
- Побудуйте орграф, що відповідає відношенню еквівалентності, яке задане на трьохелементній множині з єдиним класом еквівалентності.

Задачі і вправи

- Наведіть повні доведення наслідків 4.8.1 і 4.8.2.
- Упевнітесь в тому, що двом різним деревам T і T' з n вершинами відповідають різні списки L і L' , які будувались у доведенні теореми Келі.
- Побудуйте транзитивне замикання відношення $R = \{(a, b), (a, c), (b, d), (d, a), (b, c)\}$, яке задано на множині $A = \{a, b, c, d\}$.
- Знайдіть число маршрутів довжини 3, що з'єднують вершини a і c для відношення R з вправи 3.
- Нехай дано дерево $T = (\{1, 2, 3, 4\}, \{(1, 2), (1, 3), (3, 4)\})$. Побудуйте його матрицю суміжностей і матрицю досяжностей. Які властивості має дана матриця? Чому дорівнює четвертий степінь даної матриці?
- Доведіть, що кожне дерево є двочастинним графом.
- Які дерева являють собою повні двочастинні графи?
- Побудуйте остатні дерева для графів:
 - K_5 ; б) $K_{3,3}$; в) графа Петерсена; г) платонових графів.
- Знайдіть циклічні ранги таких графів:
 - K_n ; б) $K_{m,n}$; в) N_n ; г) W_n ; д) платонових графів;
- графа Петерсена; ж) зв'язного регулярного графа степеня k і порядку n .

10. Побудуйте остаточні дерева і асоційовані з ними фундаментальні системи циклів і розрізів таких графів:

а) K_6 ; б) $K_{3,4}$; в) W_5 ; г) C_6 ; д) платонових графів.

11. Побудуйте всі неізольовані між собою дерева шостого порядку.

ЧАСТИНА II

МАТЕМАТИЧНІ МОДЕЛІ

Розділ 5

АБСТРАКТНА ТЕОРІЯ АВТОМАТИВ

У цьому розділі вводяться основні поняття абстрактної теорії автоматів, а також основні види автоматів. Мета даного розділу — показати, що вивчення поведінки автоматів в абстрактній теорії автоматів можна звести до вивчення представлення подій в одному спеціальному класі автоматів, які називаються *автоматами Мура*.

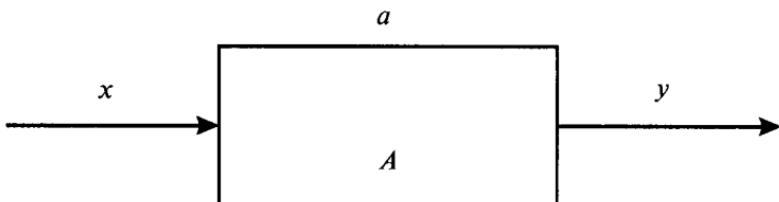
§ 5.1. ОСНОВНІ ОЗНАЧЕННЯ

1. ОЗНАЧЕННЯ АВТОМАТА І ЙОГО РІЗНОВИДИ

П'ятірка (A, X, Y, f_A, g_A) називається *автоматом Мілі*, якщо вона складається із множини станів автомата A , множини вхідних символів X , множини вихідних символів Y , функції переходів $f_A: A \times X \rightarrow A$ і функції виходів $g_A: A \times X \rightarrow Y$. Множини X і Y називають відповідно вхідним і вихідним алфавітами автомата. Як правило, автомат позначають символом множини його станів, тобто $A = (A, X, Y, f_A, g_A)$. Коли відомо, про який автомат іде мова, то індекс A в символах функцій переходів і виходів опускають.

Якщо $f(a, x) = a'$, то кажуть, що автомат A під дією вхідного сигналу $x \in X$ переходить в стан a' , або сигнал x переводить автомат A із стану a в стан a' . Якщо $g(a, x) = y$, то говорять, що автомат A перетворює в стані a вхідний сигнал $x \in X$ у вихідний сигнал $y \in Y$.

Автомат зручно задавати у вигляді нижчеприведеної схеми, де прямокутник A символізує множину станів автомата, а стрілки — канали, за допомогою яких автомат обмінюється інформацією з зовнішнім середовищем. Стрілка з позначкою x відповідає його входу, а стрілка з позначкою y — його виходу. Символи x і y , якими позначені стрілки, означають відповідно вхідний і вихід-



ний символи, а символ a над прямокутником — внутрішній стан автомата A в момент надходження символа x .

Функціонування такої схеми можна описати так. На вхід автомата подається сигнал (символ) $x \in X$, під дією якого відповідно з поточним внутрішнім станом автомата і функціїй переходу і виходу змінюється стан автомата, і на його виході з'являється вихідний сигнал (символ) $y \in Y$. Коли на вхід автомата подавати слово (послідовність вхідних символів), на виході також з'являється слово (послідовність вихідних символів). У цьому випадку автомат A можна розглядати як алфавітний перетворювач інформації, який перетворює слова напівгрупи $F(X)$ в слова напівгрупи $F(Y)$.

Послідовність вхідних сигналів можна розглядати як функцію натуального аргументу — дискретного автоматного часу. Це дає підстави трактувати автомат як дискретну динамічну систему, яка змінює свій стан у часі під дією зовнішніх і внутрішніх факторів.

Автомат A називається *ініціальним*, якщо в множині його станів виділено деякий стан, який називається *початковим*. При функціонуванні ініціального автомата вважається, що в початковий момент часу (перед подачею на його вхід деякого слова $p \in F(X)$) автомат перебуває в початковому стані.

Автомат $A = (A, X, Y, f, g)$ називається *скінченим*, якщо всі три множини — A , X і Y — скінченні, і *некінченим*, якщо хоч одна з них нескінчена.

Автомат називається *повним* або *повністю визначенім*, якщо функції переходів і виходів повністю визначені, і *частковим*, якщо хоч одна з них часткова.

Інколи трапляються automati, в яких компонент f — не функція, а деяке відношення, тобто в таких автоматах не виконується умова однозначності переходу. Автомати такого типу називаються *недетермінованими*. Якщо ж відношення f є функцією, то автомат називається *детермінованим*. Отже, для детермінованого автомата A з початковим станом a однозначно знаходиться стан b , в який автомат перейде під дією слова $p \in F(X)$. А в недетермінованому автоматі таких станів може бути не один, а кілька.

Ясно, що клас детермінованих автоматів є підкласом недетермінованих автоматів.

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

Автомат без пам'яті — це трійка (X, Y, g) , де $g: X \rightarrow Y$. Цей автомат виконує посимвольну трансформацію символів вхідного алфавіту X в символи вихідного алфавіту Y , при якому на один і той же вхідний символ $x \in X$ він реагує одним і тим же вихідним символом $y \in Y$. Поняття **автомат без пам'яті** — основне в теорії перемикальних схем, яка вивчає способи подання таких автоматів мережами із логічних елементів.

Автономний автомат — це четверка (A, Y, f, g) , де $f: A \rightarrow A$, $g: A \rightarrow Y$. Для такого автомата поданням початкового стану визначається весь подальший процес його функціонування.

Автомат без виходів, або X -автомат — це трійка (A, X, f) , де $f: A \times X \rightarrow A$. Часто X -автомат задають за допомогою четверки $(A, X, f, F \subseteq A)$, а ініціальний X -автомат — за допомогою п'ятірки $(A, X, f, \{a_0\}, F \subseteq A)$, де F — деяка підмножина станів автомата, елементи якої називаються **заключними** або **представляючими станами**, а $a_0 \in A$ — **початковий стан автомата**.

Із перелічених класів автоматів тільки X -автомати, з точки зору теорії, викликають інтерес, оскільки в двох попередніх випадках поведінка автомата наперед визначена.

Автомати Мура є окремим випадком автоматів Мілі. Автомат (A, X, Y, f, g) називається **автоматом Мура**, якщо його функція виходів $g(a, x)$ виражається функцією переходів $f(a, x)$ за допомогою рівняння $g(a, x) = h(f(a, x))$, де $h: A \rightarrow Y$. Функція h називається **функцією відміток автомата**, а її значення $h(a)$ на стані a — **відміткою цього стану**.

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

Перейдемо до розгляду скінчених автоматів і способів їх подання.

У загальному випадку для подання скінчених автоматів користуються двома стандартними способами, які називаються **універсальними** — це таблиці переходів і виходів та графи переходів і виходів.

2. ТАБЛИЦІ ПЕРЕХОДІВ І ВИХОДІВ

Таблиці переходів і виходів автомата — це дві матриці однакової розмірності, рядки яких відмічені різними символами вхідного алфавіту, а стовпчики — різними символами станів автомата. Для функції переходів (перша матриця) на перетині i -го рядка, відміченого символом $x \in X$, і j -го стовпчика, відміченого станом $a \in A$, знаходиться значення $f(a, x) = b$. Аналогічно для функції виходів (друга матриця) — на перетині i -го рядка і j -го стовпчика знаходиться значення $g(a, x) = y \in Y$. У випадку ініціального автомата вважають, що початковий стан автомата відмічає перший стовпчик як в таблиці переходів, так і в таблиці виходів.

Приклад 5.1.1

1. Автомат $A = (\{0, 1, 2, 3\}, X = \{a, b\}, Y = \{0, 1\}, \{0\}, f, g)$ задається такими таблицями переходів і виходів:

f	0	1	2	3
a	2	3	0	1
b	1	0	3	2

g	0	1	2	3
a	0	0	0	0
b	1	1	1	1

Наведений автомат переписує будь-яке слово в алфавіті $\{a, b\}$ в слово в алфавіті $\{0, 1\}$.

2. Розглянемо більш змістовний приклад автомата. Нехай $A = (\{0, 1, 2, 3\}, X = \{a = (0, 0), b = (0, 1), c = (1, 0), d = (1, 1)\}, Y = \{0, 1\}, \{0\}, f, g)$, де f і g задані такими таблицями переходів і виходів:

f	0	1	2	3
a	0	2	0	2
b	0	1	0	1
c	0	1	0	1
d	3	1	3	1

g	0	1	2	3
a	0	1	0	1
b	1	0	1	0
c	1	0	1	0
d	0	1	0	1

Неважко перевірити, що цей автомат виконує додавання двох чисел у двійковій системі числення, які поступають на його вхід у порядку зростання розрядів. Крім того, він виконує контроль знакового розряду. *

3. ГРАФ ПЕРЕХОДІВ І ВИХОДІВ

Граф переходів і виходів автомата являє собою орієнтований граф, вершини якого взаємно однозначно відповідають станам автомата. Дуги графа відмічені парами символів: перший — символом вхідного алфавіту, другий — символом вихідного алфавіту.

У цьому разі відповідність між графом переходів і виходів і функціями переходу f і виходу g така, що коли $f(a, x) = a'$ і $g(a, x) = y$, то в графі з вершини a у вершину a' веде дуга, відмічена парою (x/y) , і навпаки, коли в графі переходів і виходів автомата існує дуга (a, a') , відмічена парою (x/y) , то для функцій переходу f і виходу g виконуються рівності $f(a, x) = a'$ і $g(a, x) = y$.

З наведеної відповідності графа автомата до його функцій переходу і виходу випливає, що не всякий граф, дуги якого відмічені парами символів із деяких алфавітів X і Y , буде являти собою граф переходів і виходів автомата. Для того щоб відмічений граф був графом автомата, необхідно і достатньо, щоб виконувались дві умови, які називаються **умовами автоматності графа**:

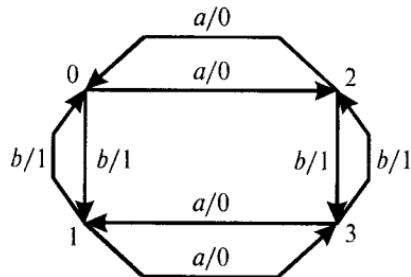
1) не існує двох дуг з одинаковими вхідними відмітками, що виходять з однієї і тієї ж вершини (умова однозначності);

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

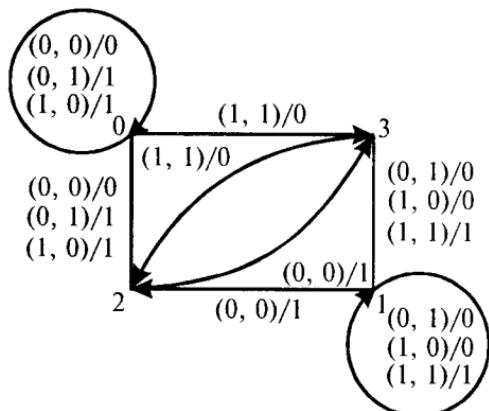
Отже, для недетермінованих автоматів не виконується умова 1), а для часткових автоматів — умова 2). Але, незважаючи на це, таблиці переходів як недетермінованих, так і часткових автоматів теж зображують графами.

Приклад 5.1.2

1. Граф переходів і виходів автомата $A = (\{0, 1, 2, 3\}, X = \{a, b\}, Y = \{0, 1\}, \{0\}, f, g)$ з прикладу 5.1.1 п.1 має вигляд:



2. Граф переходів і виходів автомата $A = (\{0, 1, 2, 3\}, X = \{a = (0, 0), b = (0, 1), c = (1, 0), d = (1, 1)\}, Y = \{0, 1\}, \{0\}, f, g)$ з прикладу 5.1.1 п.2 має такий вигляд:



При невеликому числі станів зручнішим є графовий спосіб подання автомата. Крім того, подання автоматів графом переходів і виходів дає можливість дати означення ще одного класу автоматів — **ациклічних**.

Автомат, граф переходів і виходів якого ациклічний, називається **ациклічним автоматом**.

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

$$\begin{aligned} f(a, e) &= a, f(a, px) = f(f(a, p), x), \\ g(a, e) &= e, g(a, px) = g(a, p) g(f(a, p), x). \end{aligned}$$

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

$$\begin{aligned} f(a, pq) &= f(f(a, p), q), \\ g(a, pq) &= g(a, p) g(f(a, p), q). \end{aligned}$$

Дійсно, якщо $q = x$, тобто $l(q) = 1$, то базис індукції має місце за означенням.

Якщо ж $q = q'x$ і для q' співвідношення асоціативності виконується, то

$$\begin{aligned} f(a, pq) &= f(f(a, pq'), x)) = f(f(a, p), q'x), \\ g(a, pq'x) &= g(a, pq) g(f(a, pq'), x) = \\ &= g(a, p) g(f(a, p), q') g(f(a, pq'), x) = g(a, p) g(f(a, p), q'x). \end{aligned}$$

Умовимося надалі вважати, що автомат A переходить із стану a в стан a' під дією слова p , або що слово p переводить автомат A із стану a в стан a' , якщо $f(a, p) = a'$.

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

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

4. ПІДАВТОМАТИ. ГОМОМОРФІЗМИ АВТОМАТІВ

Підмножина $B \subseteq A$ множини станів автомата A становить *підавтоматом автомата* A , якщо для всякого стану $b \in B$ і будь-якого вхідного символу $x \in X$ стан $f(b, x)$ належить множині B , причому функції переходу і виходу автомата B збігаються з функціями переходу і виходу автомата A на множині B .

З наведеного означення випливає, що основна умова, яку повинен задовольняти підавтомат, — це замкнутість множини його станів за відношенням досяжності. Іншими словами, якщо підмножина B множини станів деякого автомата A являє собою множину станів підавтомата, то ця підмножина повинна містити всі стани, досяжні з будь-якого стану цієї підмножини. В окремому випадку, якщо зафіксувати деякий стан b автомата A і побудувати множину B усіх станів, досяжних із цього стану, то ця множина разом з відповідно обмеженими функціями переходу і виходу автомата буде підавтоматом автомата A .

Нехай задано деякий скінчений автомат $A = (A, X, Y, f, g)$ і його стан $a \in A$. Відображення f_a будемо називати *відображенням, яке індукується станом a автомата* A , якщо для всякого $p \in F(X)$

$$f_a(p) = g(a, p).$$

Будемо вважати, що автомат A задає відображення h його вхідної напівгрупи на вихідну напівгрупу, якщо існує такий стан a цього автомата, що $h = f_a$.

Множиною відображень, заданих автомatom A , назовемо сукупність $\{f_a\}$ ($a \in A$) всіх попарно різних відображень.

Якщо ж автомат A ініціальний, обмежимося розглядом відображення, яке індукується його початковим станом. Це відображення будемо називати *відображенням, яке індукується автоматом* A .

Автомати A і B , які мають один і той же вхідний і вихідний алфавіт і індукують одну і ту ж сукупність відображень, тобто $\{f_a\}(a \in A) = \{f_b\}(b \in B)$, називаються *еквівалентними автоматами* ($A \sim B$).

Якщо автомати A і B ініціальні, то $A \sim B$, коли $f_{a_0} = f_{b_0}$.

Два стани a і b одного і того ж автомата A , чи двох різних автоматів A і B називаються *еквівалентними станами* ($a \sim b$), якщо $f_a = f_b$.

Безпосередньо з цих означень випливає така теорема.

Теорема 5.1.1. *Автомати $A \sim B$ тоді і тільки тоді, коли для всякого стану $a \in A$ знайдеться еквівалентний йому стан $b \in B$, і навпаки.*

Автомат називається *приведеним*, якщо всі його стани попарно нееквівалентні.

Для скінчених автоматів *приведеність* означає, що кількість станів приведеного автомата не перевищує кількості станів будь-якого іншого автомата, еквівалентного йому.

В теорії автоматів задача побудови приведеного автомата для заданого автомата є однією з найважливіших і має назву *задача мінімізації автоматів*.

Нехай дано два автомати: $A = (A, X, Y, f, g)$, $B = (B, X', Y', f', g')$ і три відображення: $z: A \rightarrow B$, $u: X \rightarrow X'$, $v: Y \rightarrow Y'$. Говорять, що відображення z , u , v *гомоморфно відображають автомат A в автомат B*, якщо $(\forall a \in A) (\forall x \in X)$ мають місце такі співвідношення:

- 1) $z(f(a, x)) = f'(z(a), u(x))$;
- 2) $v(g(a, x)) = g'(z(a), u(x))$.

Якщо ж автомати A і B ініціальні, то до умов 1) і 2) додається ще одна умова:

- 3) $z(a_0) = b_0$, де $a_0 \in A$, $b_0 \in B$ — початкові стани автоматів.

Якщо такі відображення z , u , v існують, то говорять також, що вони *реалізують гомоморфізм автомата A в автомат B*. Гомоморфізм автомата A в автомат B називається *ізоморфізмом*, якщо всі три відображення z , u , v , що реалізують цей гомоморфізм, взаємно однозначні.

Ізоморфні автомати відрізняються один від одного лише познаками своїх входів і вихідних символів і станів. Тому коли в автоматах A і B входні чи вихідні алфавіти або і входні, і вихідні алфавіти збігаються, то вважатимемо, що відображення u чи відображення v або і u , і v — тодіжні відображення.

Надалі розглядатимемо автомати A і B , в яких входні і вихідні алфавіти збігаються, тобто $A = (A, X, Y, f, g)$ і $B = (B, X, Y, f', g')$. У цьому випадку відображення $z: A \rightarrow B$ буде називатися *гомоморфізмом* автомата A в автомат B , якщо $(\forall a \in A) (\forall x \in X)$:

- 1) $z(f(a, x)) = f'(z(a), x)$;
 - 2) $g(a, x) = g'(z(a), x)$;
 - 3) $z(a_0) = b_0$ — для ініціальних автоматів.
- (5.1.1)

Окремими варіантами поняття гомоморфізму є поняття *гомоморфізму “на”* і *ізоморфізму*. Гомоморфізм z називається *гомоморфізмом “на”*, якщо відображення z є відображенням “на” (див. розділ 1), і *ізоморфізмом*, якщо z — взаємно однозначний гомоморфізм.

Якщо z — гомоморфізм автомата A в автомат B , то індукцією

за довжиною слова $p \in F(X)$ можна довести, що рівності (5.1.1) виконуються і для розширених функцій переходів і виходів:

- 1) $z(f(a, p)) = f'(z(a), p);$
 - 2) $g(a, p) = g'(z(a), p).$
- (5.1.2)

З поняттям гомоморфізму автоматів тісно пов'язане поняття еквівалентності автоматів. Дійсно, рівність $g(a, p) = g'(z(a), p)$ при переході до відображеній, які індукуються станами автомата, означає

$$f_a(p) = f_{z(a)}(p). \quad (5.1.3)$$

Оскільки рівність (5.1.3) справедлива для всякого вхідного слова p , то це означає, що стани a і $z(a)$ еквівалентні. Звідси випливає, що

$$\{f_a\} (a \in A) \subseteq \{f_{z(a)}\} (z(a) \in B). \quad (5.1.4)$$

Для автоматів справедливі твердження, аналогічні твердженням 2.1.1 і 2.1.2.

Теорема 5.1.2. Якщо $z: A \rightarrow B$ — гомоморфізм автомата A в автомат B , то образ $z(A)$ автомата A при відображенії z є підавтоматом автомата B .

Д о в е д е н н я. Достатньо показати, що підмножина $z(A)$ множини станів автомата B замкнута відносно функції переходів f' . Припустимо протилежне, тобто, що існують стан $a' \in z(A)$ і деякий $x \in X$, такі, що $f'(a', x) \notin z(A)$. Оскільки $a' \in z(A)$, то існує $a \in A$, такий, що $z(a) = a'$. Але тоді за означенням гомоморфізму стан $f(a, x)$ повинен переходити в стан $f'(z(a), x) = f'(a', x)$, який, за нашим припущенням, не належить множині $z(A)$. Одержанана суперечність доводить замкнутість множини $z(A)$ відносно функції переходів f' .

Теорема 5.1.3. Якщо $z: A \rightarrow B$ і $z': B \rightarrow C$ — гомоморфізми (гомоморфізми “на”, ізоморфізми) автоматів A і B , B і C , то добуток відображень $z * z': A \rightarrow C$ — теж гомоморфізм (гомоморфізм “на”, ізоморфізм).

Д о в е д е н н я пропонується як проста вправа.

Теорема 5.1.4. Якщо $z: A \rightarrow B$ — гомоморфізм автомата A на автомат B , то $A \sim B$.

Д о в е д е н н я. Оскільки z — гомоморфізм “на”, то для всякого стану b із B прообраз $z^{-1}(b) \neq \emptyset$. Значить, для всякого стану b із B знайдеться (хоча б один) стан a із A , такий, що $a \sim b$. За теоремою 5.1.1 і формулою (5.1.4) дана теорема справедлива.

Зауважимо, що твердження, обернене до теореми 5.1.4, хибне і для гомоморфізмів “на”, і для ізоморфізмів.

Наслідок 5.1.1. Якщо $z: A \rightarrow B$ — гомоморфізм автомата A в автомат B , то $A \sim z(A)$, де $z(A)$ — гомоморфний образ A в B .

Ядром гомоморфізму $z: A \rightarrow B$ ($\ker(z)$) називається таке розбиття множини станів автомата A на класи, при якому в один і той же клас попадають ті і тільки ті стани, які при гомоморфізмі z мають один і той же образ. Іншими словами, ядро гомоморфізму $\ker(z) = \{K(a) \mid a \in A\}$, де $K(a)$ — клас розбиття, якому належить стан a і $b \in K(a) \Leftrightarrow z(a) = z(b)$. Оскільки $\ker(z)$ — розбиття множини станів, то в силу леми 1.1.1 з ядром гомоморфізму зв'язується відношення еквівалентності R_z , таке, що $a R_z b \Leftrightarrow z(a) = z(b)$.

Безпосередньо з означення гомоморфізму автоматів випливає таке твердження.

Твердження 5.1.1. Якщо $z: A \rightarrow B$ — гомоморфізм автомата A в автомат B і $\ker(z)$ — його ядро, то:

- $a R_z b \Rightarrow a \sim b$;
- $(\forall x \in X) a R_z b \Rightarrow f(a, x) R_z f(b, x) \& g(a, x) = g(b, x)$.

Доведення пропонується як пристра вправа.

Відношення еквівалентності R на множині станів автомата A будемо називати **конгруентністю**, якщо $a R b \Rightarrow f(a, x) R f(b, x) \& g(a, x) = g(b, x)$.

Неважко перевірити, що відношення R_z і \sim — конгруентності. Для відношення R_z це випливає прямо з твердження 5.1.1, б). Покажемо, що відношення \sim — конгруентність.

Нехай $x \in X$, $p \in F(X)$, $f(a, x) = a'$, $f(b, x) = b'$ і $a \sim b$. Відношення $a \sim b$ означає, що $f_a(xp) = f_b(xp)$ і $f_a(x) = f_b(x)$, або $f_a(xp) = f_a(x)f'_a(p) = f_b(x)f'_b(p) = f_b(xp)$. Звідси випливає, що $f'_a(p) = f'_b(p)$. Оскільки слово p довільне, то це значить, що $f(a, x) = a' \sim b' = f(b, x)$, а разом з рівнянням $f_a(x) = f_b(x)$, і те, що відношення \sim — конгруентність.

Таким чином, всякий гомоморфізм автоматів A і B задає деяке відношення конгруентності на множині станів автомата A . Лема 1.1.2 підказує нам таке питання: а чи справедливе обернене твердження, тобто, чи можна за відношенням конгруентності на множині станів автомата A вказати такий автомат B , на який автомат A буде гомоморфно відображатись і при цьому ядро гомоморфізму збігатиметься з даною конгруентністю. Виявляється, що такий факт має місце.

Нехай $A = (A, X, Y, f, g)$ — автомат і R_z — відношення конгруентності на множині A . Введемо новий автомат $\bar{A} = (\bar{A}, X, Y, \bar{f}, \bar{g})$, станами якого є класи еквівалентності за відношенням R_z , а \bar{f} і \bar{g} визначаються нижче за допомогою функцій f і g таким

чином. Нехай \bar{a} — представник класу еквівалентності $K(a)$, яко-му належить стан a . Тоді

$$\bar{f}(\bar{a}, x) = \overline{f(a, x)} \text{ і } \bar{g}(\bar{a}, x) = g(a, x).$$

Оскільки R_z — конгруентність, вибір представника в класі еквівалентності $K(a)$ не впливає на задання автомата \bar{A} . Автомат \bar{A} , побудований таким чином, називається **фактор-автоматом** автомата A щодо конгруентності R_z і позначається A/R_z .

Розглянемо відображення h_z , яке ставить у відповідність кожному стану a автомата A клас еквівалентності стану $a - \bar{a} = K(a)$ — автомата A/R_z . Неважко впевнитися, що h_z гомоморфізм. Дійсно, за визначенням h_z маємо

$$h_z(f(a, x)) = \bar{b} = \overline{f(h_z(a), x)} = \bar{f}(\bar{a}, x),$$

$$h_z(g(a, x)) = \bar{g}(h_z(a), x) = \bar{g}(\bar{a}, x) = g(a, x).$$

Гомоморфізм h_z в теорії автоматів називається **канонічним**. Для канонічного гомоморфізму автоматів має місце аналог теореми про гомоморфізми (див. § 2.1).

Теорема 5.1.5. *Нехай $z: A \rightarrow B$ — гомоморфізм автомата A в автомат B і $\ker(z)$ — його ядро; тоді автомати $A/\ker(z)$ і $z(A)$ ізоморфні.*

Доведення пропонується як вправа.

5. ТЕОРЕМА ПРО ПРИВЕДЕНИЙ АВТОМАТ

Теорема про приведений автомат. *Нехай A — довільний автомат і $K(A)$ — клас всіх автоматів, еквівалентних автомату A . Тоді в класі $K(A)$ існує приведений автомат, який є гомоморфним образом будь-якого іншого автомата з класу $K(A)$ і визначається з точністю до ізоморфізму.*

Доведення. Побудуємо за автоматом A і відношенням \sim фактор-автомат A/\sim . Покажемо, що автомат A/\sim має всі властивості, які вказані в умові теореми.

Покажемо насамперед, що автомат A/\sim — приведений. Нехай \bar{a}, \bar{a}' — пара еквівалентних станів автомата A/\sim . Тоді за властивістю канонічного гомоморфізму (теорема 5.1.5) маємо, що всякі два прообрази станів \bar{a} і \bar{a}' — стани a і a' еквівалентні між собою. Але це означає, що a і a' належать одному і тому ж класу еквівалентності за відношенням \sim і при канонічному гомо-

морфізмі $A \rightarrow A/\sim$ переходять в один і той же стан автомата A/\sim , тобто $\bar{a} = \bar{a}'$. Отже, стани автомата A/\sim попарно нееквівалентні, а це означає, що автомат A/\sim — приведений.

Нехай B — довільний автомат із $K(A)$. Тоді існує гомоморфізм z автомата B на автомат A/\sim , який задається так:

$$z(b) = \bar{a} = K(a),$$

де a — деякий стан автомата A , еквівалентний стану b автомата B . Таке задання відображення z коректне в тому плані, що не залежить від вибору елемента a : якщо $a \sim b$ і $a' \sim b$, то $\bar{a} = \bar{a}'$ — їх спільний образ при канонічному гомоморфізмі. Відображення z — гомоморфізм “на”, оскільки для довільного стану $\bar{a} \in A/\sim$ існує еквівалентний йому стан a автомата A , а тоді через те, що $A \sim B$, знайдеться стан $b \in B$, такий, що $a \sim b$ (теорема 5.1.1), і який при відображені z буде переведений у стан \bar{a} .

Покажемо тепер, що z — гомоморфізм. Нехай $b \in B$ — довільний стан автомата B і $a \in A$, такий, що $a \sim b$. Тоді

$$\begin{aligned} z(f(b, x)) &= \overline{f(a, x)} = \overline{f}(\bar{a}, x) = \overline{f}(z(b), x), \\ g(b, x) &= f_b(x) = f_a(x) = g(a, x) = \overline{g}(z(b), x). \end{aligned}$$

Нехай $B \in K(A)$, в якого всі стани попарно нееквівалентні. Покажемо, що в цьому випадку гомоморфізм $z: B \rightarrow A/\sim$ буде взаємно однозначним, тобто буде ізоморфізмом. Нехай $b, b' \in B$ і $z(b) = z(b')$. Але тоді, оскільки $b \sim z(b)$ і $b' \sim z(b')$, то $b \sim b'$, а це через те, що B приведений, означає, що $b = b'$.

Теорема доведена.

З цієї теореми випливає, що для одержання приведеного автомата, еквівалентного заданому, достатньо побудувати максимальне розбиття множини станів даного автомата на класи еквівалентних між собою станів. Опишемо метод побудови такого розбиття, який ґрунтуються на понятті *i-еквівалентності станів* ($i = 1, 2, \dots$).

Стани a і a' називаються *i-еквівалентними*, якщо ($\forall p \in F(X)$) довжини i маємо $g(a, p) = g(a', p)$. *1-еквівалентними* є стани, які мають однакові стовпці в таблиці виходів. Якщо стани a і a' не *i-еквівалентні*, то вони називаються *i-різними*. Очевидно, що *i-еквівалентність* станів — це звичайне відношення еквівалентності. Класи еквівалентності даного відношення називаються *i-класами*. З умов автоматності випливає, що для всякого значення i справедлива нерівність $(i+1)$ -класи $\leq i$ -класів, тобто якщо стани $(i+1)$ -еквівалентні, то вони і *i-еквівалентні*.

Стани називаються **∞ -еквівалентними**, якщо для всякого i ($i \in \{1, 2, \dots\}$) вони i -еквівалентні. Це відношення також є відношенням еквівалентності, а його класи еквівалентності будемо називати **∞ -класами**.

Теорема 5.1.6. *Два стани a і a' еквівалентні тоді і тільки тоді, коли вони ∞ -еквівалентні.*

Доведення випливає з означення відношення еквівалентності станів автомата і означення ∞ -еквівалентності.

Для побудови приведеного автомата розглянемо такий, взагалі кажучи, нескінчений процес розбиття множини станів на ∞ -класи.

На першому етапі з таблиці виходів автомата знаходимо розбиття множини станів автомата на 1-класи. Далі, вважаючи, що система i -класів автомата уже побудована, застосовуємо операцію розщеплення i -класів для побудови $(i+1)$ -класів. Ця операція визначається так.

Нехай K_1, K_2, \dots, K_m — деяке розбиття множини станів автомата A на класи. Тоді розщепленням цього розбиття будемо називати таке розбиття K'_1, K'_2, \dots, K'_n множини станів цього автомата, яке задовольняє такі умови:

а) кожна із множин K'_1, K'_2, \dots, K'_n цілком належить одній із множин K_1, K_2, \dots, K_m ;

б) $(\forall x \in X) (\forall a \in K'_j) f(a, x) \in K_i$, де K_i визначається символом x і класом K'_j і є одним із i -класів;

в) якщо $K'_i, K'_j \subseteq K_l$ ($i \neq j$), то $(\exists x \in X) (\forall a \in K'_i) (\forall a' \in K'_j) f(a, x) \in K_p, f(a', x) \in K_q, p \neq q$.

Для деякого розбиття операція розщеплення класів не приводить до їх зміни, наприклад, коли кожний клас — це окремий стан автомата. Застосовуючи операцію розщеплення класів, одержуємо нове розбиття, класи якого є підкласами відповідних класів попереднього розбиття. При цьому два стани a і a' належать одному і тому ж $(i+1)$ -класу, якщо $(\forall x \in X) f(a, x) \text{ і } f(a', x)$ належать одному і тому ж i -класу.

Припустимо тепер, що попереднім для операції розщеплення класів було розбиття множини станів на i -класи. Тоді з рівності $g(a, xp) = g(a, x)g(f(a, x), p)$, де p — вхідне слово довжини i , випливає, що стани a і a' , які попали при розщепленні деякого i -класу в один і той же його підклас, будуть $(i+1)$ -еквівалентними. Дійсно, оскільки a і a' лежать в одному i -класі і $i \geq 1$, то $g(a, x) = g(a', x)$, а оскільки $f(a, x)$ і $f(a', x)$ також лежать в одному і тому ж i -класі, то і $g(f(a, x), p) = g(f(a', x), p)$. Навпаки, якщо стани a і a' лежать у різних класах розбиття, яке одержане в результаті застосування операції розщеплення, то це може бути

тільки тоді, коли вони вже лежали в різних i -класах (значить, не були навіть i -еквівалентними) або, в протилежному випадку, для деякого x стани $f(a, x)$ і $f(a', x)$ лежать у різних i -класах. А це значить, що існує таке слово p довжини i , що $g(a, xp) \neq g(f(a', x), p)$.

Зауважимо, нарешті, що коли початковий автомат скінчений, то через деякий час послідовність розбиття множини його станів на класи, що одержуються на різних етапах процесу розбиття, стабілізується і являє собою розбиття на ∞ -класи. Дійсно, на кожному кроці i ($i \geq 1$) число i -класів не перевищує числа станів самого автомата, але тоді в силу скінченності автомата знайдеться таке значення i , що всі $(i+1)$ -класи будуть збігатися з i -класами. Для такого розбиття індукцією за довжиною входного слова p легко показати, що яким би не було слово p для двох довільних i -еквівалентних станів a і a' , буде виконуватися рівність $g(a, p) = g(a', p)$.

Контрольні питання

1. Дайте означення автомата.
2. Який автомат називається скінченим, повним, частковим, детермінованим, недетермінованим, ациклічним, автоматом без виходів?
3. Які ви знаєте способи задання автоматів?
4. Що являють собою умови автоматності графа?
5. Дайте означення відображення, яке представлене в автоматі.
6. Дайте означення еквівалентності станів автомата і автоматів.

§ 5.2. ПРЕДСТАВЛЕННЯ ПОДІЙ В АВТОМАТАХ. АВТОМАТИ МУРА І НЕДЕТЕРМІНОВАНІ АВТОМАТИ

Цей невеликий за обсягом параграф сuto теоретичний. У ньому розглядаються умови і способи представлення довільних відображень і подій в автоматах, а також взаємозв'язок між автоматами Мілі і автоматами Мура і детермінованими і недетермінованими автоматами Мілі з точки зору представлення подій у цих автоматах. Встановлюється, що як детерміновані, так і недетерміновані автомати Мілі еквівалентні автоматах Мура.

1. АВТОМАТНІ ВІДОБРАЖЕННЯ

Вище було введено поняття алфавітного відображення для вивчення поведінки автомата. Розглянемо тепер умови, які повинні задовольняти алфавітне відображення, що індукується автоматом, і як побудувати алгоритми, які дають змогу синтезувати автомати, що представляють ці відображення.

Раніше для розширених функцій виходів було встановлено співвідношення

$$g(a, pq) = g(a, p) \cdot g(f(a, p), q),$$

або якщо умовитися писати скорочення ap для $f(a, p)$, то

$$g(a, pq) = g(a, p) \cdot g(ap, q).$$

Переписуючи його з використанням позначень для відображень, які індукуються станами автоматів, одержуємо співвідношення

$$z_a(pq) = z_a(p)z_{ap}(q).$$

З цієї рівності випливає дві необхідні умови — назовемо їх **умовами автоматності** відображень, яким повинно відповідати всяке відображення, що представляється в автоматах:

- 1) $l(z(p)) = l(p)$ для всякого p із $F(X)$;
- 2) якщо $p \leq q$, то $z(p) \leq z(q)$.

Відношення “ \leq ” — це відношення часткового порядку на множині $F(X)$, яке задається таким чином: $p \leq q \Leftrightarrow q = pp'$, тобто коли p — початкове підслово слова q .

Наведені умови автоматності не тільки необхідні, а й достатні для представлення відображень в автоматах. Перед тим як показати достатність цих умов, введемо деякі необхідні поняття.

Будемо говорити, що слово p' є часткою від ділення слова q на слово p , і позначати це за допомогою рівності $p' = q \parallel p$, якщо $p \leq q$ ($q = pp'$). Коли ж нерівність $p \leq q$ не виконується, то результат операції $q \parallel p$ будемо вважати невизначенним.

На множині автоматних відображень, які діють на напівгрупі слів $F(X)$, для кожного слова $p \in F(X)$ визначимо **оператор зсуву**, який перетворює відображення z у відображення z_p і задається рівнянням $z_p(q) = z(pq) \parallel z(p)$. Відображення z_p називається **зсувом відображення z** .

З другої умови автоматності відображення і нерівності $p \leq pq$ випливає, що для всякого слова p із $F(X)$ оператор зсуву $[]_p$ повністю визначений на множині автоматних відображень.

Для довільного слова p і довільного автоматного відображення z зсув $z_p = [z]_p$ теж буде автоматним відображенням. Дійсно,

$$l(z_p(q)) = l(z(pq) \parallel z(p)) = l(z(pq)) - l(z(p)) = l(pq) - l(p) = l(q).$$

Далі, нехай $q \leq r$; тоді $pq \leq pr$ і $z(pq) \leq z(pr)$. Якщо поділимо обидві частини нерівності на одне і те ж початкове слово $z(p)$, то одержимо

$$z(pq) \parallel z(p) \leq z(pr) \parallel z(p), \text{ або } z_p(q) \leq z_p(r).$$

Таким чином, із $q \leq r$ випливає, що $z_p(q) \leq z_p(r)$, тобто друга умова автоматності теж виконується.

З означення оператора зсуву легко одержати співвідношення $[[\cdot]]_p|_q = [\cdot]_{pq}$, яке виконується для будь-якої пари слів із $F(X)$. З цього співвідношення випливає, що множину операторів зсуву можна розглядати як напівгрупу, і ця напівгрупа ізоморфна напівгрупі слів $F(X)$.

Розглянемо тепер два способи синтезу автомата за відображенням.

Перший спосіб. Візьмемо за множину станів автомата A_z , який необхідно побудувати, множину $F(X)$ — множину всіх слів в алфавіті X разом з пустим словом e . Функції переходів і виходів задамо рівняннями $f(p, x) = px$; $g(p, x) = z_p(x)$.

Покажемо індукцією за довжиною вхідного слова, що стан e автомата A_z індукує відображення z . Для слів, які складаються з однієї букви, рівність $h_e(p) = z(p)$ очевидна. Дійсно,

$$h_e(x) = g(e, x) = z_e(x) = z(x) \parallel z(e) = z(x).$$

Нехай тепер це справедливо для слова p довжини, більшої 1, тобто $h_e(p) = z(p)$. Тоді для слова px маємо $h_e(px) = h_e(p)z_p(x) = z(p)(z(px) \parallel z(p)) = z(px)$.

Зауважимо, що для конкретного автоматного відображення z можуть існувати такі слова p і q , що їх зсуви z_p і z_q збігаються.

Другий спосіб. Візьмемо множину всіх зсувів $\{z_p\}$ ($p \in X$) відображення z за множину станів автомата A_z , який необхідно синтезувати, а функції переходів і виходів автомата задамо рівняннями

$$f(z_p, x) = z_{px}; \quad g(z_p, x) = z_p(x).$$

Пропонується довести індукцією за довжиною вхідного слова, що стан z_e — початковий стан автомата A_z — індукує відображення z .

Із способів побудови автоматів A_z і $A_{\bar{z}}$ випливає така теорема.

Теорема 5.2.1. *Будь-яке відображення, яке задовольняє умови автоматності, може бути представлене в автоматі. Якщо такий автомат приведений, то число його станів визначається числом зсувів даного відображення.*

З цієї теореми і означення оператора зсуву одержуємо такий наслідок.

Наслідок 5.2.1. Разом зі всяким відображенням, яке представлене в даному автоматі, у цьому автоматі також представлений і всякий його зсув.

Дійсно, якщо відображення h_a , індуковане станом a , збігається з відображенням z і z_p — зсув відображення z , то відображення z_p і h_{ap} збігаються:

$$z_p(q) = z(pq) \parallel z(p) = h_a(pq) \parallel h_a(p) = (h_a(p)h_{ap}(q)) \parallel h_a(p) = h_{ap}(q).$$

Оцінюючи автомати, побудовані обома способами за одним і тим же автоматним відображенням, можна сказати, що вони являють собою дві можливі крайності: при першому способі побудови стани автомата вводяться скрізь, де тільки можливо, а при другому — максимально економно, тобто лише там, де це абсолютно необхідно. У першому випадку діаграма переходів і виходів побудованого автомата незалежно від початкового відображення являє собою нескінченнє дерево, а в другому — одержаний автомат приведений, і число його станів не залежить від початкового відображення.

Таким чином, відображення, яке не задовольняє умови автоматності, в автоматі представити неможливо. Правда, існує простий спосіб, за допомогою якого можна побудувати інше відображення — автоматне і за яким однозначно можна знайти початкове відображення.

Цей спосіб ґрунтуються на розширенні алфавітів X і Y : до алфавіту X добавляється буква u , а до алфавіту Y — буква v . Шукане відображення \bar{z} буде так: якщо $z(p) = q$, то

$$\bar{z}(pu \dots u) = \underbrace{v \dots v}_{l(q) \text{ раз}} q$$

Визначаючи так відображення z на всіх можливих словах із множини $F = \{pu^{l(z(p))} \mid p \in F(X)\}$, довизначимо його на всіх початкових відрізках цих слів так, щоб виконувались умови автоматності. Таке визначення коректне, оскільки воно не приводить до неоднозначності відображення. Якщо буква u ні разу не зустрічається в слові $r = q \parallel p$, то образом $z(r)$ цього слова буде слово $\underbrace{vv \dots v}_{l(r) \text{ раз}}$. Якщо ж буква u зустрічається в слові r , то вона

однозначно визначає слово p і, отже, образ $z(r)$. Всяке слово, яке не належить множині F і не є початковим відрізком слова з F , можна записати у вигляді p_1p_2 , де p_1 — слово максимальної довжини з множини F або початковий відрізок деякого слова з F . У цьому випадку відображення \bar{z} на слові p_1p_2 визначимо за допо-

могою рівняння $\bar{z}(p_1 p_2) = \bar{z}(p_1) \underbrace{vv \dots v}_{l(p_2) \text{ раз}}.$ Побудоване таким чином відображення \bar{z} автоматне.

Легко показати, що за відображенням z однозначно відновлюється початкове відображення $\bar{z}.$ Дійсно, якщо приписати до слова p таке число символів $u,$ щоб образ цього слова мав вигляд $v \dots vqv,$ то після викреслювання символів v одержимо слово $q,$ яке за побудовою відображення \bar{z} і буде образом слова p при відображені $z.$ Із сказаного випливає справедливість такого твердження.

Теорема 5.2.2. Для довільного алфавітного відображення $f: F(X) \rightarrow F(Y)$ існує автоматне відображення $g: F(X) \cup \{u\} \rightarrow F(Y) \cup \{v\},$ яке однозначно визначає початкове відображення $f.$

Описаний спосіб побудови відображення \bar{z} називається **стандартним способом вирівнювання довжин слів.** Але цей спосіб не завжди найекономніший спосіб перетворення довільного алфавітного відображення в автоматне. У цьому легко переконатися, якщо його застосувати до автоматного відображення. Через те на практиці, як правило, вдаються до послідовного дописування букв u і v (праворуч від входного слова і ліворуч від вихідного), щоразу перевіряючи виконання умов автоматності.

2. АВТОМАТНІ СИСТЕМИ ПОДІЙ

Нехай X — деякий, можливо і нескінчений, алфавіт. Всяка підмножина $S \subseteq F(X)$ називається **подією** в алфавіті $X.$ Автоматною системою подій в алфавіті X будемо називати всяке сімейство $\{S_i\}$ ($i \in I$, де I — деяка множина індексів) непустих подій у цьому алфавіті, таке, що

$$\bigcup_{i=1} S_i = F(X) \setminus \{e\} \text{ і } S_i \cap S_j = \emptyset, \text{ якщо } i \neq j.$$

Між автоматними системами подій в алфавіті X і автоматними відображеннями з напівгрупи $F(X)$ в деяку іншу напівгрупу $F(Y)$ можна встановити однозначну відповідність. Нехай z — деяке автоматне відображення $F(X)$ в $F(Y),$ і нехай алфавіт Y такий, що для кожної його букви знайдеться хоча б одне слово $p \in F(X),$ для якого ця буква входить в образ $z(p).$ Інакше, з алфавіту Y можна виключити всі букви, які не задовільняють цю умову. Тепер для всякого $y \in Y$ позначимо S_y подією, яка складається із всіх слів p із $F(X),$ для яких $z(p)$ закінчується символом

у. Сукупність $\{S_y\}$ ($y \in Y$) таких подій, очевидно, буде автоматою в алфавіті X . Побудовану таким чином по відображенням з систему автоматних подій будемо надалі називати **канонічною системою подій** автоматного відображення z з вхідною напівгрупою $F(X)$ і вихідною напівгрупою $F(Y)$.

Для канонічної системи подій автоматного відображення з справедлива і обернена властивість — вона однозначно визначає відображення z . Дійсно, нехай $p = x_{i_1} x_{i_2} \dots x_{i_n}$ — довільне слово із $F(X)$. Припустимо, що слово $x_{i_1} x_{i_2} \dots x_{i_j}$ (для $j = 1, 2, \dots, n$) належить події S_y . Тоді за другою умовою автоматності і визначення подій $\{S_y\}$ ($y \in Y$) образом слова p при відображення z може бути лише слово $y_1 y_2 \dots y_n$. Це слово отримане виключно за сукупністю подій $\{S_y\}$ без безпосереднього звертання до відображення z . Якщо індекси автоматної системи подій явно не задані, то можна будувати відображення, для якого ця система була б канонічною з точністю до позначення букв вихідного алфавіту. Отже, має місце така теорема.

Теорема 5.2.3. *Канонічна система подій довільного автоматного відображення z : $F(X) \rightarrow F(Y)$ є автоматною системою подій в алфавіті X . Навпаки, всяка автоматна система подій в алфавіті X може розглядатися як канонічна система подій деякого автоматного відображення з вхідною напівгрупою $F(X)$. При цьому відображення z визначається з точністю до позначок букв вихідного алфавіту.*

Ця теорема дає можливість звести вивчення автоматних відображень до вивчення автоматних систем подій. Зокрема, задача представлення відображень в автоматах може бути зведена до задачі представлення подій в автоматах.

Говорять, що подія S_y в алфавіті Y **представлена в ініціальному автоматі** A вихідним сигналом y , якщо вхідним алфавітом автомата A є алфавіт X , а S_y складається із всіх тих і тільки тих слів в алфавіті X , образи яких при відображення z індукуються автоматом A , закінчуються буквою y .

Автоматна система подій називається **представленою в ініціальному автоматі** A , якщо кожна подія цієї системи представлена в автоматі A деяким вихідним сигналом.

Виходячи з теореми 5.2.3, всяку автоматну систему подій $\{S_i\}$ ($i \in I$) можна розглядати як канонічну систему подій деякого автоматного відображення. Побудувавши ініціальний автомат, який індукує це відображення, представимо початкову автоматну систему подій вихідними символами побудованого таким чином автомата.

Для того щоб подію S в алфавіті X представити в автоматі,

необхідно розглянути систему подій $\{S, F(X) \setminus (S \cup \{e\})\}$, яка, очевидно, є автоматною системою. Отже, подію S можна представити в автоматі. З усього сказаного випливає таке твердження.

Теорема 5.2.4. *Всяка подія може бути представлена вихідним сигналом деякого, взагалі кажучи, нескінченного ініціального автомата Мілі. Всяка автоматна система подій допускає представлення в ініціальному автоматі Мілі.*

3. АВТОМАТИ МУРА

Вище говорилось, що $X - Y$ -автомат називається автоматом Мура, якщо його функція виходів виражається функцією переходів $f(a, x)$ за допомогою рівняння $g(a, x) = z(f(a, x))$, де z — **функція відміток** станів. **Відміткою** стану a називається значення функції $z(a)$ на цьому стані.

Хоча автомати Мура являють собою окремий випадок автоматів Мілі, але з точки зору індукованих цими автоматами відображені вони виявляються не біднішими, ніж автомати Мілі.

Теорема 5.2.5. *Для всякого автомата Мілі $A = (A, X, Y, f_A, g_A)$ існує еквівалентний йому автомат Мура B .*

Д о в е д е н н я. Візьмемо за множину станів автомата B множину $A \times Y$ і задамо функції переходів і виходів такими правилами.

Якщо $g_A(a, x) = y'$ і $f_A(a, x) = a'$, то покладаємо

$$f_B((a, y), x) = (a', y'), g_B((a, y), x) = g_A(a, x) = y'.$$

З такого визначення функцій f_B і g_B видно, що коли на множині $A \times Y$ ввести функцію відміток z за допомогою рівності $z((a, y)) = y$, то для функції g_B буде виконуватись співвідношення

$$g_B((a, y), x) = z(f_B((a, y), x)),$$

яке означає, що автомат B є автоматом Мура.

Тепер індукцією за довжиною вхідного слова покажемо, що стани a і (a, y) (y — довільний символ із Y) індукують одне і те ж відображення. Нехай $p = x_{i_1} x_{i_2} \dots x_{i_n}$ — довільне вхідне слово, і нехай $a_{j_1}, a_{j_2}, \dots, a_{j_{n+1}}$ — послідовність станів автомата A , така, що $a_{j_1} = a$ і $a_{j_{k+1}} = f_A(a_{j_k}, x_{i_k})$ для $j = 1, \dots, n$.

Внаслідок рівності $g_B((a, y), x) = g_A(a, x)$ для слів, які складаються з однієї букви, образи відображень u_a і $u_{(a, y)}$ збігаються. Допускаючи, що це справедливо для слова p довжини n , покажемо, що образи цих відображень збігаються і для слова xp більшої довжини. Виходячи з властивості відображень

і

$$u_a(xp) = u_a(x)u_{ax}(p)$$

$$u_{(a, y)}(xp) = u_{(a, y)}(x)u_{((a, y), x)}(p),$$

а також з означення функції переходів

$$f_B((a, y), x) = (f_A(a, x), y')$$

і нашого припущення, маємо $u_{ax}(p) = u_{(ax, y')}(p)$, а значить,

$$u_a(xp) = u_{(a, y)}(xp).$$

Теорема доведена.

Можливий і інший шлях побудови автомата B . Поставимо у відповідність кожному стану a автомата A множину $\{\bar{a}\} \cup \{(a, x) \mid x \in X\}$. Об'єднання всіх таких множин приймемо за множину станів автомата B , а функції f_B і g_B переходів і виходів цього автомата задамо за допомогою рівностей:

$$f_B(\bar{a}, x) = (a, x), f_B((a, x), x') = (f_A(a, x), x');$$

$$g_B(\bar{a}, x) = g_A(a, x), g_B((a, x), x') = g_A(f_A(a, x), x').$$

Позначимо через z функцію відміток, яка станові (a, x) автомата B ставить у відповідність відмітку $g_A(a, x)$. Тепер справедливи рівності:

$$g_B(\bar{a}, x) = z(f_B(\bar{a}, x)); g_B((a, x), x') = z(f_B((a, x), x')).$$

Отже, побудований автомат є автоматом Мура. Індукцією за довжиною входного слова p можна показати, що для всякого $x \in X$ стани \bar{a} і a еквівалентні і стани ax і (a, x) також еквівалентні.

Обидва способи побудови автомата B за автомatem A мають право на існування, оскільки якщо автомат A скінчений, то і автомати Мура, побудовані обома способами, теж будуть скінченними, але в залежності від числа символів в алфавітах X і Y матимуть різне число станів. В одних випадках економніший автомат дає перший спосіб побудови, а в інших — другий спосіб побудови.

Наслідок 5.2.2. Якщо автомат Мілі A скінчений і має n станів, то існує еквівалентний йому автомат Мура B з $k \cdot n$ станами, де k — число символів вихідного алфавіту.

Наслідок 5.2.3. Якщо автомат Мілі A скінчений і має n станів, то існує еквівалентний йому автомат Мура B з $(m + 1)n$ числом станів, де m — число символів вихідного алфавіту.

Автомати Мура, як і автомати Мілі, можна задавати таблицями переходів і виходів. Але для автоматів Мура існує і ком-

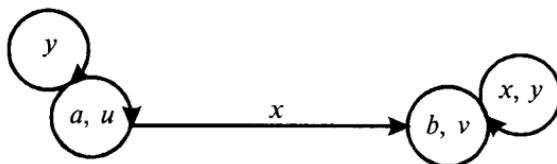
пактніший спосіб задання. Дійсно, функція відміток автомата Мура однозначно задається функціями переходів і виходів для всіх тих станів, які є значеннями функції переходів. Для всіх останніх станів автомата функція $z(a)$ може бути невизначеною. Надалі умовимося довизначати функцію $z(a)$ на всю множину станів автомата Мура і задавати автомат не функціями переходів і виходів, а функціями переходів і відміток. При необхідності функцію виходів можна визначати за допомогою рівності $g(a, x) = z(f(a, x))$.

Для автоматів Мура функція переходів задається вже відомими таблицями переходів або відмітками дуг на діаграмах переходів. Функцію відміток можна задавати простіше, ніж функцію переходів. Для цього, як правило, станам автомата Мура в таблицях переходів ставлять у відповідність певні відмітки або у випадку діаграм позначають вершини діаграми цими відмітками.

Приклад 5.2.1

Таблиця переходів і відміток автомата Мура

Відмітка	Стан	x	y
u	a	b	a
v	b	b	b



Діаграма переходів і відміток. ▲

При роботі з автоматами Мура слід мати на увазі, що хоча підавтомат автомата Мура й ізоморфний образ автомата Мура теж будуть автоматами Мура, властивість автоматів бути автоматами Мура не зберігається, зокрема при гомоморфізмах автоматів. Звідси випливає, що автомат, еквівалентний автомату Мура, може не бути автоматом Мура, а алгоритм побудови приведеного автомата може давати автомат, який не є автоматом Мура. У зв'язку з цим в теорії автоматів Мура вводяться спеціальні поняття: мурівська еквівалентність і мурівський гомоморфізм.

Стани a і a' одного або двох різних автоматів Мура називаються **мурівськи еквівалентними**, якщо вони мають одну і ту ж відмітку і еквівалентні у звичайному розумінні. Два автомати

мурівськи еквівалентні, якщо для кожного стану одного автомата знайдеться мурівськи еквівалентний йому стан іншого автомата, і навпаки.

Гомоморфізм $h: A \rightarrow A'$ автомата A в автомат A' називається **мурівським**, якщо він зберігає функцію відміток, тобто для всяко-го $a \in A$ справедливе $z'(h(a)) = z(a)$.

Автомат Мура називається **мурівським приведеним**, якщо всі його стани мурівськи попарно нееквівалентні між собою.

Для автоматів Мура мають місце аналоги теореми 5.1.4 і тео-реми про приведений автомат.

Теорема 5.2.6. Якщо $h: A \rightarrow B$ — мурівський гомоморфізм автомата A на автомат B , то автомати A і B мурівськи еквівалентні.

Доведення випливає з означення.

Теорема 5.2.7 (про приведений автомат Мура). У класі $K(A)$ всіх автоматів, мурівськи еквівалентних заданому автомату A , існує один і з точністю до мурівського ізоморфізму лише один приведений автомат, на який мурівськи гомоморфно відображаються всі автомати з цього класу.

Для автоматів Мура, в яких вихідні сигнали збігаються з від-мітками станів, зручніше говорити про представлення подій множинами станів, а не вихідними сигналами.

Будемо вважати, що подія S в алфавіті X представлена в іні-ціальному автоматі Мура A множиною його станів F , якщо вхід-ний алфавіт автомата A збігається з X , а S складається з тих і тільки тих слів, які переводять автомат A із початкового стану в стани з множини F . Слід зауважити, що коли початковий стан належить до множини заключних станів F , то подія, яка пред-ставляється множиною F , буде включати пусте слово, чого не могло бути при представленні подій вихідними сигналами.

Теорема 5.2.8. Якщо подія S представлена в ініціальному автоматі Мура вихідним сигналом u , то подія, яка можливо поповнена лише пустим словом, представляється в тому ж автоматі множиною всіх станів, які відмічені вихідним сигналом u . Якщо подія S представлена в ініціальному автоматі множиною станів F , то, визначаючи функцію відміток так, щоб вона приймала значення u на всіх станах множини F і тільки на таких станах, одержимо ініціальний автомат Мура, в якому подія S , окрім пустого слова (якщо $a_0 \notin F$), представляється вихідним сигналом u .

Доведення. Нехай F — множина всіх станів автомата A , які мають відмітку u , а $S(a_0, F)$ — множина всіх слів, які перево-дять автомат A з початкового стану a_0 в один із станів множини F . Нехай rx — довільне слово, таке, що слово $g(a_0, rx)$ закін-чується символом u (позначимо S_u подію, яка складається із всіх

таких слів px). Тоді внаслідок рівності $g(f(a_0, p), x) = z(f(a_0, px))$ можна стверджувати, що $f(a_0, px) \in F$ і $px \in S(a_0, F)$, тобто $S_y \subseteq S(a_0, F)$.

Нехай тепер $px \in S(a_0, F)$; тоді $g(f(a_0, p), x) = z(f(a_0, px)) = y$ і, значить, слово $px \in S_y$. Таким чином, події S_y і $S(a_0, F)$ рівні між собою з точністю до пустого слова e .

Теорема доведена.

Теорема 5.2.8 дозволяє обмежитися розглядом подій, які представляються в автоматах множинами станів. Функції виходів при цьому не відіграють ніякої ролі і тому надалі, розглядаючи події, які представляються в автоматах, будемо розглядати автомати без виходів, або X -автомати.

Підводячи короткий підсумок, підкреслимо, що, розглядаючи детерміновані автомати без виходів з точки зору представлення подій множиною станів таких автоматів, ми не обмежуємо загальності розгляду.

4. НЕДЕТЕРМІНОВАНІ АВТОМАТИ

Недетермінованим автоматом (недетермінованим $X-Y$ -автоматом) називається четвірка (A, X, Y, R) , де A, X, Y — відповідно множина станів, вхідний і вихідний алфавіти автомата, а $R \subseteq A \times X \times Y \times A$ — його відношення переходів і виходів.

Тепер очевидно, що коли подати відношення R як $\{(a, x, g(a, x), f(a, x))\}$, то детермінований автомат можна розглядати як недетермінований. **Ініціальний недетермінований автомат** — це недетермінований автомат з виділеним початковим станом a_0 , який задається у вигляді п'ятірки (A, X, Y, R, a_0) . Якщо $(a, x, y, a') \in R$, то говорять, що вхідний символ x може перевести автомат A із стану a в стан a' і видати на виході символ y . Недетерміновані автомати, як і детерміновані, зручно задавати за допомогою діаграм переходів і виходів. Недетермінований $X-Y$ -автомат називається частковим, якщо існує такий стан a і такий вхідний символ x , що відношення R не має жодної четвірки, в якої перші два компоненти дорівнюють a і x .

Якщо алфавіт Y складається з єдиного вихідного символу, то це окремий випадок недетермінованих автоматів — недетерміновані автомати без виходів. Ці автомати являють собою трійки виду (A, X, R) , де A, X — множини станів і вхідний алфавіт, а $R \subseteq A \times X \times A$ — відношення переходів.

Аналогічно розширеним функціям переходів і виходів детермінованих автоматів можна розглядати розширені відношення переходів і виходів недетермінованих автоматів. Відношення

$R^* \subseteq A \times F(X) \times F(Y) \times A$ називається *розширенням відношення переходів і виходів R*, якщо для довільної четвірки $(a, x_1x_2\dots x_n, y_1y_2\dots y_m, a') \in R^*$ виконується рівність $m = n$ і існує така послідовність станів a_1, a_2, \dots, a_{n+1} , що $a_1 = a$, $(a_i, x_i, y_i, a_{i+1}) \in R$ для $i = 1, 2, \dots, n$ і $a_{n+1} = a'$.

Для недетермінованих автоматів без виходів вводиться розширення функція переходів $Q^*: A \times F(X) \rightarrow B(A)$, де $B(A)$ — булеван множини A , яка також називається *розширенням функції переходів Q*: $A \times X \rightarrow B(A)$ і задається таким чином: для кожного слова $p = x_1x_2\dots x_n \in F(X)$ і $a' \in Q^*(a, p)$ можна вказати таку послідовність станів a_1, a_2, \dots, a_{n+1} , що $a = a_1$, $a_{i+1} \in Q(a_i, x_i)$ для $i = 1, 2, \dots, n$ і $a_{n+1} = a'$.

Ініціальним настроєнім недетермінованим X-автоматом (автоматом без виходів) (A, X, Q, a_0, F) називається *X-автомат* (A, X, Q) , в якого стан a_0 взятий за початковий, а $F \subseteq A$ взято за множину заключних (фінальних) станів. Подія S називається *подією*, що представлена в автоматі (A, X, Q, a_0, F) , коли вона складається з тих і тільки тих слів p , для яких $Q^*(a_0, p)$ містить хоча б один стан із F .

Ініціальні недетерміновані *X-автомати A і B* називаються *еквівалентними*, якщо для всякої підмножини \bar{A} із A знайдеться така підмножина \bar{B} із B , що події, які представлені в настроєніх *X-автоматах* $(A, X, Q_A, a_0, \bar{A})$ і $(B, X, Q_B, b_0, \bar{B})$, збігаються, і на впаки.

Два стани a і a' недетермінованого *X-автомата* (A, X, Q) називаються *еквівалентними*, якщо при виборі довільної підмножини F із A як множини заключних станів настроєні *X-автомати* (A, X, Q, a, F) і (A, X, Q, a', F) представляють одну і ту ж подію.

З теоретичної точки зору досить важливим є твердження, яке встановлює рівносильність детермінованих і недетермінованих автоматах при представленні подій в скінчених автоматах.

Теорема про детермінізацію. Для всякого ініціального недетермінованого *X-автомата* A існує еквівалентний йому детермінований ініціальний *X-автомат* B , причому якщо автомат має n станів, то автомат B може бути побудований так, що число його станів не перевищує величини 2^n .

Д о в е д е н н я. Нехай (A, X, Q_A, a_0) — недетермінований *X-автомат* і $B(A)$ — булеван множини станів A . На $B(A)$ введемо відношення досяжності: множина A'' досяжна з множини A' , якщо існує слово $p \in F(X)$, таке, що

$$A'' = \bigcup_{a \in A'} Q^*(a, p).$$

За множину станів шуканого детермінованого X -автомата B візьмемо множину всіх станів, які досяжні з множини $A_0 = \{a_0\}$. Множина $\{a_0\}$ буде служити початковим станом автомата B . Функцію переходів автомата B представимо за допомогою рівності

$$f_B(\bar{A}, x) = \bigcup_{a \in \bar{A}} Q(a, x),$$

де \bar{A} — деякий стан автомата B .

Нехай \bar{A} — довільна підмножина із A , і нехай \bar{B} — всі підмножини A , які досяжні із $\{a_0\}$, і такі, що їх перетин з \bar{A} непустий. Покажемо включенням в обидва боки, що події, представлені в X -автоматах $(A, X, Q_A, a_0, \bar{A})$ і $(B, X, Q_B, \{a_0\}, \bar{B})$, збігаються.

Нехай слово $p = x_1x_2\dots x_n$, таке, що $Q_A(a_0, p) \cap \bar{A} \neq \emptyset$. Це означає, що існує послідовність станів $a_0, a_1, \dots, a_n = a'$, така, що $a_{i+1} \in Q_A(a_i, x_{i+1})$ для $i = 0, 1, \dots, n - 1$. Але тоді множина, яка представляє стан $f_B(\{a_0\}, p)$, повинна включати стан a' , а значить, a' має належати до \bar{B} .

Навпаки, нехай для $p = x_1x_2\dots x_n$ стан $f_B(\{a_0\}, p)$ належить множині \bar{B} . Це значить, що множина, яка представляє цей стан, включає деякий елемент a' із \bar{A} . Тоді з означення досяжності множин і способу задання функції Q^* індукцією за довжиною слова p можна побудувати послідовність $a' = a_n, a_{n-1}, \dots, a_0$, таку, що $a_i \in f_B(a_0, x_1x_2\dots x_i)$, а це означає при $i = n$, що $Q_A^*(a_0, p) \cap \bar{A} \neq \emptyset$.

На завершення доведення цієї теореми зауважимо, що множина, яка складається з n елементів, має 2^n підмножин і, значить, число станів автомата B не перевищує 2^n .

Важливо те, що оцінка 2^n є точною верхньою оцінкою [24]. Отже, переход від недетермінованого автомата до відповідного детермінованого дає експоненціальний ріст числа станів. З практичної точки зору це може значно ускладнювати роботу, тому часто користуються недетермінованими автоматаами, оскільки для представлення подій ця модель вимагає набагато менших за числом станів автоматів.

Задачі та вправи

1. Дайте повні доведення теорем 5.2.3, 5.2.4.
2. Дайте повне доведення теореми 5.1.5.

3. Покажіть, що для відображень z і h , які задані рівностями $z(p) = pp$ (подвоєння слова p), $h(p) = p'$, де p' одержано із p вилученням всіх входжень деякої букви вхідного алфавіту, існує економніша процедура вирівнювання довжини слів, ніж стандартна.

4. Побудуйте граф і таблицю переходів для скінченного автомата, який представляє:

- а) всі парні числа в двійковій системі числення;
- б) всі непарні числа в двійковій системі числення.

5. Нехай $X = \{0, 1\}$, а S — множина всіх слів із $F(X)$, які в двійковій системі дають парні числа, і S' — слова із $F(X)$, які в двійковій системі числення дають непарні числа. Чи буде $\{S, S'\}$ автоматною системою подій?

Р о з д і л 6

ТЕОРІЯ СКІНЧЕННИХ АВТОМАТІВ

У даному розділі розглядаються основні положення теорії скінченних автоматів. Наводяться ефективні алгоритми синтезу та аналізу скінченних автоматів, а також алгоритми їх мінімізації.

§ 6.1. ПОДІЇ, АЛГЕБРА РЕГУЛЯРНИХ ПОДІЙ. ОСНОВНА ТЕОРЕМА ТЕОРИЇ СКІНЧЕННИХ АВТОМАТІВ

1. РЕГУЛЯРНІ ПОДІЇ

Нагадаємо, що *подією в алфавіті* X називається довільна множина S слів із напівгрупи $F(X)$. Говорять, що подія S *представлена в автоматі* B множиною заключних станів F тоді і тільки тоді, коли $(\forall p \in S) f(a, p) \in F$. Пара $S(U(X), W)$ називається *алгеброю подій* в алфавіті X , якщо її носієм $U(X)$ є множина всіх подій в алфавіті X , а W містить дві бінарні операції: *диз'юнкцію* і *множення*, і одну унарну — *ітерацію*.

Диз'юнкцією $S \vee S_1$ двох подій S і S_1 називається теоретико-множинне об'єднання подій S і S_1 , тобто $S \vee S_1 = S \cup S_1$.

Добутком $S * S_1$ подій S і S_1 називається подія, яка складається із всіх слів виду pq , де p, q — довільні слова із S і S_1 відповідно, тобто $S * S_1 = \{pq \mid p \in S, q \in S_1\}$. Наприклад, всяке слово виду $x_1x_2\dots x_k$ можна представити у вигляді добутку слів, які складаються з однієї букви: $(x_1) * (x_2) * \dots * (x_k)$. Звідси випливає, що перестановка множників в операції добутку подій, загалом, може не зберігати результату, тобто ця операція некомутативна. Очевидно, що коли операції диз'юнкції та добутку застосувати до скінченних подій, то в результаті одержимо знову скінченні події.

Ітерацією $\{S\}$ події S називається подія, яка складається із пустого слова e і всіх можливих слів виду $p_1 p_2 \dots p_k$, де p_1, p_2, \dots, p_k — довільні слова із S , а k — довільне натуральне число із \mathbb{N}^+ . Основна властивість операції ітерації полягає в тому, що вона дає можливість будувати нескінченні події із скінченних. Наприклад, ітерація події $x \in X$ дає нескінченну множину слів

$$\{e, x, x^2, \dots, x^n, \dots\}.$$

Пуста множина слів називається **неможливою подією**. З означення операції ітерації випливає, що ітерація пустого слова і неможливої події є пустим словом, тобто $\{e\} = \{\emptyset\} = e$.

Елементарною подією в алфавіті $X = \{x_1, x_2, \dots, x_m\}$ називається одна з $m + 1$ одноелементних подій e, x_1, x_2, \dots, x_m , де e — пусте слово. Всяка подія, яку можна одержати з елементарних застосуванням будь-якого скінченного числа операцій диз'юнкції, добутку й ітерації, називається **регулярною подією**, або **регулярною множиною слів**, а всяке її задання через елементарні події три вказаних операції — **регулярним виразом**.

Безпосередньо з означення операцій алгебри регулярних подій випливають такі тотожності:

- A1) $S \vee T = T \vee S; S \vee S = S;$
- A2) $S \vee (T \vee F) = (S \vee T) \vee F; S * (T * F) = (S * T) * F;$
- A3) $(S \vee T) * F = S * F \vee T * F; S * (T \vee F) = S * T \vee S * F;$
- A4) $\{\{S\}\} = \{S\}; \{S\} = e \vee S * \{S\}; S * \{S\} = \{S\} * S;$
- A5) $\{\{S\} \{T\}\} = \{S \vee T\}; \{S\} * \{S\} = \{S\};$
- $\{S\} \vee S = \{S\}; \{e\} = e;$
- A6) $S * e = e * S = S; S * \emptyset = \emptyset * S = \emptyset; S \vee \emptyset = S; \{\emptyset\} = e.$

Інколи в алгебрі подій розглядають і інші операції, які не належать до числа основних. Найчастіше вводяться унарна операція **доповнення** і бінарна операція **перетину подій**.

Доповненням S' події S в алфавіті X називається множина всіх слів із $F(X)$, які не входять в S .

Перетином $S \cap S_1$ подій S і S_1 називається подія, яка складається із всіх слів, що одночасно входять в обидві події — S і S_1 .

Зауважимо, що коли операція перетину подій замкнута на множині всіх скінченних подій, то операція доповнення не є такою на цій множині.

Розглядаючи регулярні вирази, умовимося називати **одночленом** регулярний вираз, в якому операція диз'юнкції не є останньою із виконуваних операцій. Диз'юнкція двох чи більшого числа регулярних виразів називається **многочленом** алгебри регулярних виразів. Наприклад, одночленами є вирази $\{S\}$, ST , $(S \vee \vee T)Q$, а вирази $(S \vee QR \vee R\{S\})$ і $(ST \vee SR)$ будуть многочленами: перший складається з трьох одночленів, а другий — з двох.

Слід зазначити, що вирази $ST \vee SR$ і $S(T \vee R)$ представляють одну і ту ж подію, хоча перший із них — многочлен, а другий — одночлен.

Циклічною глибиною регулярного виразу називається довжина його максимальної послідовності підпорядкованих одна одній ітерацій. При цьому перший знак ітерації вважається підпорядкованим другому, якщо перший входить до виразу, який знаходиться під знаком другої ітерації. Наприклад, якщо регулярні вирази P, Q, R мають циклічну глибину нуль, то циклічною глибиною виразів $\{P\} \vee P\{Q\} \vee R$ буде одиниця, $\{\{P\}S\}$ — два і $P \vee \vee RS$ — нуль.

Циклічна глибина регулярної події вводиться як найменша циклічна глибина регулярних виразів, які представляють дану подію. Зокрема, регулярні вирази $\{\{x\}\{y\}\}$ і $\{(x \vee y)\}$ представляють одну і ту ж регулярну подію, але мають різну циклічну глибину, і тому циклічна глибина цієї події відповідає другому, а не першому виразу. Те, що ця подія не може мати глибину нуль, випливає з такого простого твердження.

Твердження 6.1.1. *Скінченні події, і тільки вони, мають нульову циклічну глибину.*

Доведення пропонується як пристра вправа (див. вправу 1 в кінці параграфа).

Поняття циклічної глибини дає можливість упорядковувати події за складністю. Подія S складніша, ніж подія T , якщо циклічна глибина першої більша, ніж циклічна глибина другої.

2. ТЕОРЕМА АНАЛІЗУ СКІНЧЕННИХ АВТОМАТІВ

Між скінченними автоматами і регулярними множинами існує тісний зв'язок, який виражається таким твердженням.

Теорема 6.1.1 (основна теорема теорії скінченних автоматів). *Клас регулярних подій збігається з класом скінченно-автоматних подій [8].*

Доведення цієї теореми проводиться шляхом побудови двох алгоритмів: *алгоритму аналізу*, який будує регулярний вираз для події, що представлена в даному автоматі, і *алгоритму синтезу*, який будує за регулярним виразом скінчений ініціальний автомат, що представляє задану цим виразом подію деякою множиною своїх станів. Часто твердження про включення регулярних подій у скінченно-автоматні події, і навпаки, виділяють як окремі і називають відповідно *теоремами аналізу і синтезу*.

Теорема аналізу скінченних автоматів. *Нехай $A = (A, X, f, a_0, F)$ — скінчений налаштований X -автомат з множиною станів $\{a_0, F\}$*

$a_1, \dots, a_n\}$, а S_{ij} означає подію, яка складається із всіх таких слів в алфавіті X , що переводять автомат A із стану a_i в стан a_j . Для побудови регулярного виразу події, представленого в автоматі A , внаслідок рівності $S(a_0, F) = \bigcup_{a_j \in F} S_{0j}$ достатньо побудувати регулярний вираз для події S_{ij} ($i, j = 0, 1, 2, \dots, n$).

Для всякого слова $p = x_1x_2\dots x_n \in S_{ij}$ множину станів $\{f(a_i, x_1), f(a_i, x_1x_2), \dots, f(a_i, x_1x_2\dots x_{n-1})\}$ будемо називати **проміжними** при переході автомата A із стану a_i в стан a_j під дією слова p . Нехай S_{ij}^k ($i, j = 1, 2, \dots, n; k = 0, 1, \dots, n$) означає подію, яка складається з тих і тільки тих слів, що переводять автомат A із стану a_i в стан a_j і використовують при цьому стани a_1, a_2, \dots, a_k як проміжні. Зокрема, якщо $k = 0$, то S_{ij}^0 означає подію, яка складається з тих слів, що переводять автомат A із стану a_i безпосередньо в стан a_j (тобто без будь-яких проміжних станів). Звідси випливає, що S_{ij}^0 можна записати у вигляді

$$S_{ij}^0 = \bigvee_{f(a_i, x)=a_j} x \vee S_{ij}, \quad \text{де } S_{ij} = \begin{cases} e, & \text{якщо } i = j; \\ \emptyset, & \text{якщо } i \neq j. \end{cases}$$

Отже, S_{ij}^0 — подія регулярна, причому регулярний вираз може бути знайдений безпосередньо за таблицею переходів (чи діаграмою переходів) автомата.

Припустимо, що регулярність всіх подій S_{ij}^m , в яких верхній індекс набуває значення $0, 1, \dots, k-1$, уже доведена.

У даному випадку регулярні вирази для подій S_{ij}^k ($i, j = 1, 2, \dots, n$) можна знайти, користуючись рекурентними співвідношеннями

$$S_{ij}^k = S_{ij}^{k-1} \vee S_{ik}^{k-1} \{S_{kk}^{k-1}\} S_{kj}^{k-1} \quad (i, j = 1, 2, \dots, n). \quad (6.1.1)$$

Дійсно, з виразу (6.1.1) випливає, що слова, з яких складається права частина i -го співвідношення, переводять автомат A із стану a_i в стан a_j і використовують при цьому стани a_1, a_2, \dots, a_k як проміжні, тобто

$$S_{ij}^k = S_{ij}^{k-1} \vee S_{ik}^{k-1} \{S_{kk}^{k-1}\} S_{kj}^{k-1}.$$

Навпаки, нехай слово p переводить автомат A зі стану a_i в стан a_j через проміжні стани a_1, \dots, a_k . Якщо стан a_k серед проміжних станів не зустрічається, то $p \in S_{ij}^{k-1}$. У протилежному випадку слово p можна записати у вигляді $p = p_1p_2\dots p_l$, де слово p_1 переводить автомат A із стану a_i в стан a_k , кожне із слів p_2, \dots, p_{l-1} переводить автомат із стану a_k в стан a_k і слово p_l переводить автомат

із стану a_k в стан a_j , причому стан a_k при всіх цих переходах не зустрічається серед проміжних станів. А це значить, що

$$p_1 \in S_{ik}^{k-1}, p_2, p_3, \dots, p_{l-1} \in S_{kk}^{k-1}, p_l \in S_{kj}^{k-1}.$$

Таким чином, доведено, що довільне слово p із S_{ij}^k входить в $S_{ij}^{k-1} \vee S_{ik}^{k-1}\{S_{kk}^{k-1}\}S_{kj}^{k-1}$, тобто рекурентне співвідношення (6.1.1) існує.

Тепер зауважимо, що $S_{ij}^n = S_{ij}$.

Теорема доведена.

З теореми аналізу можна одержати алгоритм аналізу скінчених автоматів. Виходячи з даного автомата і співвідношення (6.1.1), цей алгоритм задає регулярний вираз для події вигляду S_{ij}^k . А регулярний вираз для події, представленою в цьому автоматі, знаходиться у вигляді диз'юнкції ($\vee S_{0j}^n$), де j пробігає всі значення станів із множини заключних станів F .

Необхідно зауважити, що при використанні цього алгоритму на практиці зовсім не обов'язково шукати регулярні вирази для всіх S_{ij}^k , а можна шукати лише для тих, які потрібні при побудові події, що представлена в автоматі. Потрібні події можна виділити, якщо рухатись “з кінця” — з події, яка нас цікавить, і спускатись до більш простих подій (в яких менше значення параметра k). Після цього, рухаючись від більш простих до більш складних подій, будуємо регулярні вирази для всіх необхідних подій. У процесі цієї побудови можна застосовувати тотожні співвідношення, які спрощують регулярні вирази. Зокрема, щоб знизити циклічну глибину виразу, необхідно застосовувати співвідношення

$$\{\emptyset\} = e \quad i \quad \{e\} = e.$$

Приклад 6.1.1

Застосуємо алгоритм аналізу до автомата A , діаграма переходів якого зображена на рис. 6.1.1. Початковим станом автомата є стан 1, а заключними станами — стани 2 і 3.

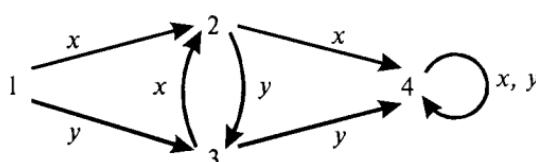


Рис. 6.1.1

Шукана подія S_A , представлена в автоматі A , очевидно, буде мати вигляд $S_A = S_{12}^4 \vee S_{13}^4$.

Застосовуючи рекурентне співвідношення (6.1.1), одержуємо

$$S_A = (S_{12}^3 \vee S_{14}^3 \{S_{44}^3\} S_{42}^3) \vee (S_{13}^3 \vee S_{14}^3 \{S_{44}^3\} S_{43}^3).$$

З діаграми переходів автомата видно, що із стану 4 не можна перейти в жодний інший стан, і тому події S_{42} і S_{43} неможливі. Внаслідок співвідношення $S * \emptyset = \emptyset$ регулярний вираз спрощується до такого вигляду:

$$S_A = S_{12}^3 \vee S_{13}^3. \quad (6.1.2)$$

Користуючись формулою (6.1.1), знаходимо вирази для S_{12}^3 і S_{13}^3 :

$$\begin{aligned} S_{12}^3 &= S_{12}^2 \vee S_{13}^2 \{S_{33}^2\} S_{32}^2; \\ S_{13}^3 &= S_{13}^2 \vee S_{12}^2 \{S_{22}^2\} S_{23}^2. \end{aligned} \quad (6.1.3)$$

Тепер будуємо вирази для S_{12}^2 , S_{13}^2 , S_{33}^2 , S_{32}^2 :

$$\begin{aligned} S_{12}^2 &= S_{12}^1 \vee S_{12}^1 \{S_{22}^1\} S_{22}^1; \\ S_{13}^2 &= S_{13}^1 \vee S_{12}^1 \{S_{22}^1\} S_{23}^1; \\ S_{33}^2 &= S_{33}^1 \vee S_{32}^1 \{S_{22}^1\} S_{23}^1; \\ S_{32}^2 &= S_{32}^1 \vee S_{32}^1 \{S_{22}^1\} S_{22}^1. \end{aligned} \quad (6.1.4)$$

Необхідно побудувати вирази для S_{12}^1 , S_{22}^1 , S_{13}^1 , S_{23}^1 , S_{32}^1 , S_{33}^1 :

$$\begin{aligned} S_{12}^1 &= S_{12}^0 \vee S_{11}^0 \{S_{11}^0\} S_{12}^0 = x \vee \emptyset \{\emptyset\} \emptyset = x; \\ S_{22}^1 &= S_{22}^0 \vee S_{21}^0 \{S_{11}^0\} S_{12}^0 = \emptyset \vee \emptyset \{\emptyset\} x = \emptyset; \\ S_{13}^1 &= S_{13}^0 \vee S_{11}^0 \{S_{11}^0\} S_{13}^0 = y \vee \emptyset \{\emptyset\} y = y; \\ S_{23}^1 &= S_{23}^0 \vee S_{21}^0 \{S_{11}^0\} S_{13}^0 = y \vee \emptyset \{\emptyset\} y = y; \\ S_{32}^1 &= S_{32}^0 \vee S_{31}^0 \{S_{11}^0\} S_{12}^0 = x \vee \emptyset \{\emptyset\} x = x; \\ S_{33}^1 &= S_{33}^0 \vee S_{31}^0 \{S_{11}^0\} S_{13}^0 = \emptyset \vee \emptyset \{\emptyset\} y = \emptyset. \end{aligned}$$

Підставляючи одержані вирази в систему (6.1.4), знаходимо:

$$\begin{aligned} S_{12}^2 &= x \vee x \{\emptyset\} \emptyset = x; \\ S_{13}^2 &= y \vee x \{\emptyset\} y = y \vee xey = y \vee xy; \\ S_{33}^2 &= \emptyset \vee x \{\emptyset\} y = \emptyset \vee xey = xy; \\ S_{32}^2 &= x \vee x \{\emptyset\} \emptyset = x \vee \emptyset = x. \end{aligned}$$

Аналогічно одержуємо вирази системи (6.1.3):

$$S_{12}^3 = x \vee (y \vee xy)\{xy\}x = e \vee (y \vee e)\{xy\}x;$$

$$S_{13}^3 = (y \vee xy) \vee (y \vee xy)\{xy\}xy = (e \vee x)y(e \vee \{xy\}).$$

Остаточно маємо

$$S_A = (e \vee (y \vee e)\{xy\})x \vee (e \vee x)y(e \vee \{xy\}). \quad \blacktriangleleft$$

3. ТЕОРЕМА СИНТЕЗУ СКІНЧЕННИХ АВТОМАТІВ

Для доведення теореми синтезу користуються теоремою про детермінізацію недетермінованого X -автомата. Для цього необхідно показати, що для всякого регулярного виразу знайдеться налаштований недетермінований X -автомат, який представляє ту ж регулярну подію, що і даний вираз. Існування такого автомата доводиться індукцією за числом операцій у початковому регулярному виразі і випливає з таких тверджень.

Лема 6.1.1. *Неможлива подія \emptyset , і всі елементарні події в алфавіті X є скінченно-автоматними.*

Доведення леми випливає з того, що ми можемо безпосередньо навести діаграми переходів автоматів $A(\emptyset)$, $A(e)$ і $A(x)$, які представляють відповідно події \emptyset , e і $x \in X$ (див. рис. 6.1.2). На рис. 6.1.2 початкові стани автоматів є ті, які знаходяться ліворуч. Заключні стани позначені зірочкою всередині відповідної вершини діаграми. Відсутність відміток у деяких дуг діаграми означає, що переходи, які відповідають цим дугам, можуть здійснюватись під дією будь-якого вхідного сигналу.

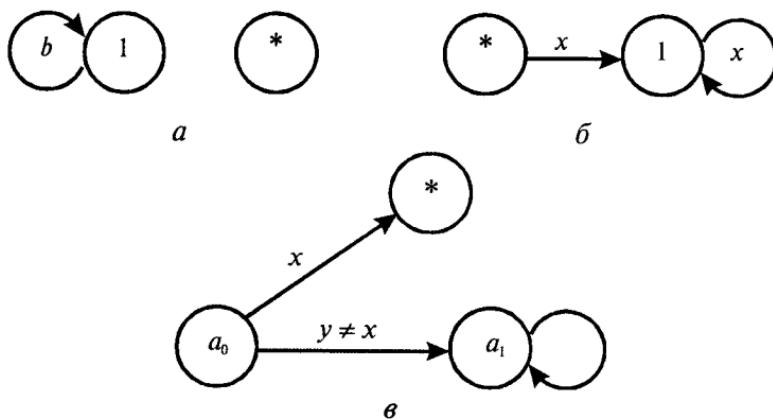


Рис. 6.1.2. Діаграми переходів автомата:
a — $A(\emptyset)$; *b* — $A(e)$; *c* — $A(x)$

Лема доведена.

Лема 6.1.2. Якщо події S і T — скінченно-автоматні, то події $S \vee T$, ST і $\{S\}$ можна представити в скінченних налаштованих недетермінованих автоматах.

Доведення цієї леми зводиться до розробки правил композиції автоматів, згідно з якими за автоматами $A(S)$ і $A(T)$ будується автомати $A(S \vee T)$, $A(ST)$ і $A(\{S\})$.

Нехай $A(S) = (A, X, f, A_0, F)$, $A(T) = (B, X, f', B_0, F')$ — автомати, які представляють події S і T відповідно і в яких A_0 , B_0 — множини початкових станів, а F і F' — множини заключних станів. Побудову автоматів $A(S \vee T)$, $A(ST)$ і $A(\{S\})$ будемо вести за такими правилами [83].

Правило 1 (побудова автомата $A(S \vee T)$). Недетермінований налаштований автомат $A(S \vee T)$, який представляє подію $S \vee T$, будується так: $A(S \vee T) = (A \cup B, X, g, A_0 \cup B_0, F \cup F')$, де функція переходів g задається за допомогою рівностей

$$g(a, x) = \begin{cases} f(a, x), & \text{якщо } a \in A(S), \\ f'(a, x), & \text{якщо } a \in A(T). \end{cases}$$

Графічне зображення цієї композиції автоматів дано на рис. 6.1.3.

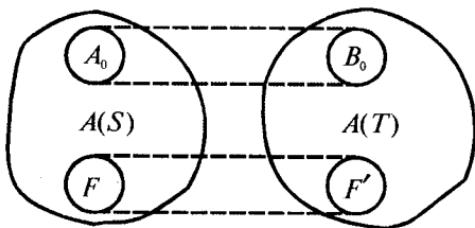


Рис. 6.1.3.

Схема побудови діаграми переходів недетермінованого автомата $A(S \vee T)$ за діаграмами переходів автоматів $A(S)$ і $A(T)$

Доведення того, що автомат $A(S \vee T)$ представляє подію $(S \vee T)$, очевидне, оскільки всяке слово, яке раніше переводило початковий стан одного із автоматів в один із заключних станів того ж автомата, тепер переводитиме цей початковий стан (як початковий стан автомата $A(S \vee T)$) в один із заключних станів (заключний стан автомата $A(S \vee T)$).

Правило 2 (побудова автомата $A(S * T)$). Автомат $A(S * T)$, який представляє подію $(S * T)$, будується так: $A(S * T) = (A \cup B, X, g, \bar{A}_0, \bar{F})$, де

$$\bar{A}_0 = \begin{cases} A_0, & \text{якщо } e \notin S, \\ A_0 \cup B_0, & \text{якщо } e \in S; \end{cases}$$

$$\bar{F} = \begin{cases} F', & \text{якщо } e \notin T, \\ F \cup F', & \text{якщо } e \in S. \end{cases}$$

$$g(a, x) = \begin{cases} f(a, x), & \text{якщо } a \in A \text{ і } a \notin F, \\ f(a, x) \cup \left(\bigcup_{a \in B_0} f'(a, x) \right), & \text{якщо } a \in F, \\ f'(a, x), & \text{якщо } a \in B; \end{cases}$$

Отже, функція переходів автомата $A(S * T)$ збігається з функціями переходів автоматів $A(S)$ і $A(T)$ для станів, які не є заключними для автомата $A(S)$. А для заключних станів автомата $A(S)$ переход під дією сигналу $x \in X$ може бути здійснений у всі стани, в які він міг бути здійснений в автоматі $A(S)$, а також у всі ті стани, в які можна попасті з будь-якого початкового стану автомата $A(T)$ під дією того ж сигналу x (наводиться схема побудови діаграми переходів автомата $A(S * T)$ за діаграмами переходів автоматів $A(S)$ і $A(T)$ у випадку, коли $e \notin S$ і $e \notin T$ (рис. 6.1.4)).

Покажемо, що побудований таким чином автомат $A(S * T)$ представляє подію $S * T$. Розглянемо випадок, коли $e \notin S$ і $e \notin T$. Нехай p — слово із $(S * T)$; тоді його можна записати у вигляді $p = p_1xp_2$, де $p_1 \in S$, $xp_2 \in T$, і вказати стани a_0 , a^* , b_0 і b^* , такі, що $f(a_0, p_1) = a^*$ і $f'(b_0, xp_2) = b^*$.

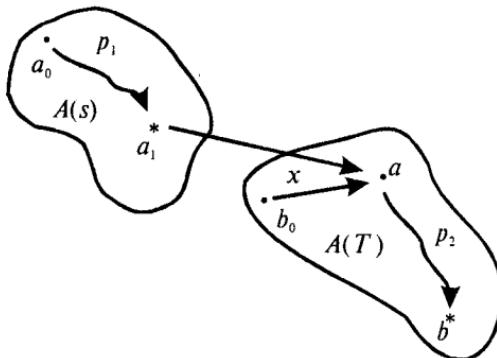


Рис. 6.1.4.

Схема побудови діаграми переходів автомата $A(S * T)$ за діаграмами переходів автоматів $A(S)$ і $A(T)$ у випадку, коли $e \notin S$ і $e \notin T$

На рис. 6.1.4, крім уже перелічених станів, виділимо стан \bar{a} , в який автомат $A(T)$ переходить із b_0 під дією першого символу слова xp_2 . З побудови автомата $A(S * T)$ випливає, що із стану a^* можна перейти в стан \bar{a} під дією символу x , але це означає, що

під дією слова $p = p_1xp_2$ (як видно з рис. 6.1.4) можна перейти із a_0 в b^* , тобто $p \in S * T$.

Навпаки, нехай p — слово, яке переводить автомат $A(S * T)$ із початкового стану a_0 в заключний стан b^* . В процесі цього переходу виділимо моменти, коли автомат $A(S * T)$ останній раз був у стані із множини A і перший раз — у стані із множини B . Стани автомата $A(S * T)$ в ці моменти позначимо a^* і \bar{a} , через x позначимо символ, який переводить автомат $A(S * T)$ із стану a^* в стан \bar{a} , через p_1 і p_2 — слова, які переводять автомат $A(S * T)$ із a_0 в a^* і з \bar{a} в b^* . Але тоді за побудовою автомата $A(S * T)$ $p_1 \in S$, $p = p_1xp_2$ і в множині B_0 знайдеться стан b_0 , такий, з якого під дією символу x можна перейти в стан \bar{a} . А це означає, що $xp_2 \in T$, а $p \in S * T$.

Випадки, коли одна з подій — S чи T , чи обидві одразу включають пусте слово e , вимагають незначних міркувань, які пов'язані з розширенням множини початкових або заключних станів, і їх розгляд пропонується як вправа (див. вправу 3 в кінці параграфа).

Правило 3 (побудова автомата $A(\{S\})$). Автомат $A(\{S\})$, який представляє подію $\{S\}$, можна одержати з автомата $A(S)$ в результаті таких дій (рис. 6.1.5):

- розширення множини станів автомата $A(S)$ станом 0, який буде служити єдиним початковим станом автомата $A(\{S\})$;
- розширення множини заключних станів за рахунок стану 0;
- розширення області значень функції переходів для заключних станів і доповнення її в стані 0 за допомогою рівностей

$$g(a, x) = \begin{cases} \bigcup_{\bar{a} \in A_0} f(\bar{a}, x), & \text{якщо } a = 0; \\ f(a, x), & \text{якщо } a \in A \text{ і } a \notin F; \\ f(a, x) \cup \left(\bigcup_{\bar{a} \in A_0} f(\bar{a}, x) \right), & \text{якщо } a \in F. \end{cases}$$

Доведемо, що побудований таким чином автомат $A(\{S\})$ представляє подію $\{S\}$. Якщо $p \in \{S\}$, тобто $p = p_1p_2...p_k$, де $p_1, p_2, \dots, p_k \in S$, то знайдуться пари (a_0^i, a_1^*) , \dots , (a_0^k, a_k^*) , перші компоненти яких є елементами з A_0 , а другі — елементами з множини F , і такі, що слова p_i ($i = 1, 2, \dots, k$) переводять автомат $A(\{S\})$ із стану a_0^i в стан a_i^* . Можна вважати, що слова p_i непусті, тобто $p_i = x_i p'_i$, $x_i \in X$. Але тоді, якщо позначити a_i ($i = 1, 2, \dots, k$) стан, в який автомат $A(\{S\})$ переходить із a_0^i під дією першого

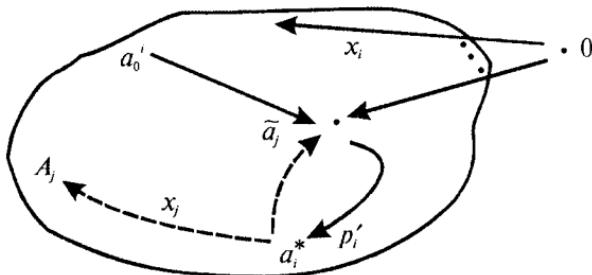


Рис. 6.1.5. Схема побудови діаграми переходів автомата $A(\{S\})$ за діаграмами переходів автомата $A(S)$: --- — додаткові переходи; $\cdot 0$ — додатковий стан, який для автомата $A(\{S\})$ є початковим станом; $x_i p_i'$ — слово події S ; a_i ($i = 1, 2, \dots, k$) — стан, в який переходить $A(S)$ з деякого початкового стану під дією деякого входного символу

символу слова p_i , то неважко переконатися (див. рис. 6.1.5), що під дією символу x_i автомат $A(\{S\})$ переїде з $\cdot 0$ в \bar{a}_i , після цього під дією p_i — в стан a_i^* , під дією x_2 — в стан \bar{a}_2 , і врешті-решт, під дією слова p_k із стану \bar{a}_k — в стан a_k^* . А це значить, що слово p є таким, яке представлено в автоматі $A(\{S\})$.

Обернене твердження про те, що слово, представлене в автоматі $A(\{S\})$, належить $\{S\}$, доводиться аналогічно. Для цього необхідно лише розбити слово p , яке переводить автомат $A(\{S\})$ із стану $\cdot 0$ в один із заключних станів, на частини відповідно до моментів часу, коли під дією наступного символу слова p автомат $A(\{S\})$ переходить із деякого заключного стану в стан, куди під дією того ж входного символу можливий переход із деякого стану, початкового для автомата $A(S)$ (див. вправу 4 в кінці параграфа).

Отже, доведення леми 6.1.2 закінчено, а разом з ним і доведення теореми синтезу і основної теореми теорії скінчених автоматів.

З наведеного вище доведення теореми синтезу легко знайти алгоритм синтезу налаштованих автоматів за даним регулярним виразом. Суть цього алгоритму полягає в послідовному застосуванні правил 1—3 відповідно до даного регулярного виразу. Розглянемо приклад.

Приклад 6.1.2

Синтезувати автомат, який представляє подію $S = (x \vee y)\{x\}$. Для синтезу автомата $A(S)$, який представляє подію S , необхідно

побудувати автомати $A(x)$ і $A(y)$, які представляють події x і y відповідно. А далі, користуючись правилами 1—3, за автоматами $A(x)$ і $A(y)$ побудувати автомати $A(x \vee y)$, $A(\{x\})$ і, нарешті, автомат $A(S)$.

Будуємо автомати $A(x)$ і $A(y)$ (рис. 6.1.6).

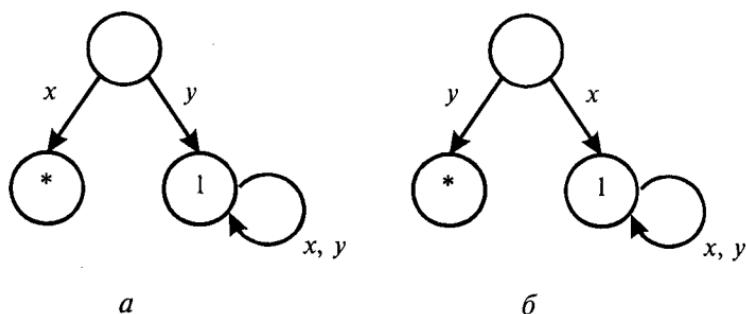


Рис. 6.1.6. Графи переходів автоматів:
а — $A(x)$; б — $A(y)$

Застосовуючи правила 1 і 3, будуємо автомати $A(x \vee y)$ і $A(\{x\})$ (рис. 6.1.7).

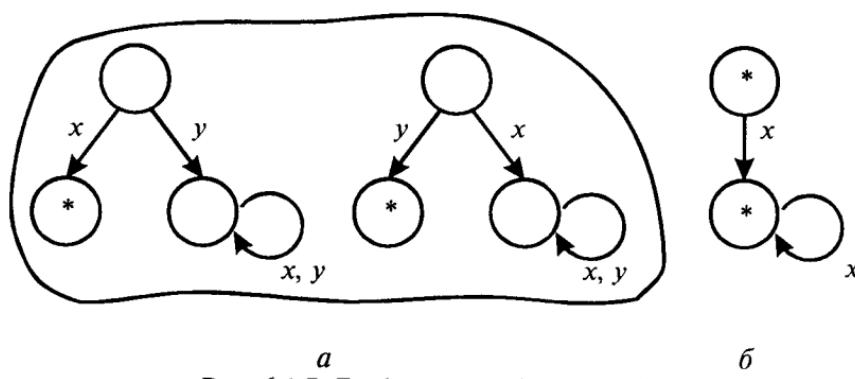


Рис. 6.1.7. Графи переходів автоматів:
а — $A(x \vee y)$; б — $A(\{x\})$

Стані, з яких недосяжні заключні стані, називають **туниками**. Зауважимо, що в процесі синтезу автомата можна вилучити тупикові стани. Подія, яка представляється в даному автоматі, очевидно, не змінюється, а число станів автомата може зменшитися, що досить суттєво для подальшого процесу детермінізації автомата і наочності діаграми переходів. Правда, автомат після вилучення тупикових станів може стати частковим.

Враховуючи це зауваження, автомати $A(x \vee y)$ і $A(\{x\})$ можна спростити (рис. 6.1.8).

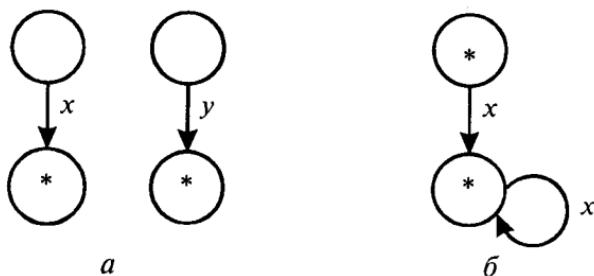


Рис. 6.1.8. Графи переходів автоматів:
 $a - A(x \vee y)$; $b - A(\{x\})$

Тепер будуємо автомат $A(S)$, користуючись правилом 2 (рис. 6.1.9).

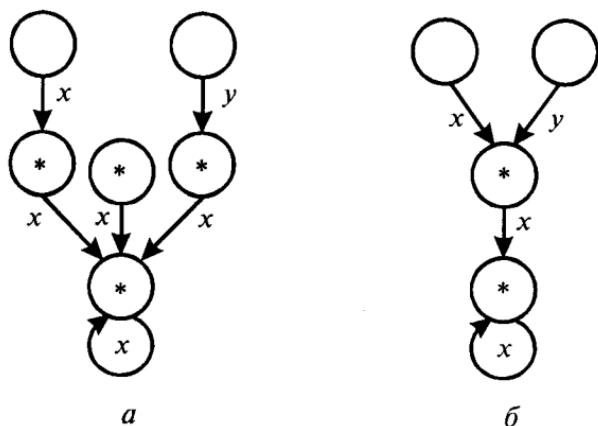


Рис. 6.1.9. Графи переходів автомата $A(S)$:
 a — із застосуванням правила 2; b — остаточний вигляд

При побудові автомата $A(S)$ були вилучені стани, які недосяжні з початкових станів автомата. Мотивація цього вилучення така ж, як і для тупикових станів.

Як уже зазначалося, автомат, одержаний в результаті синтезу, може бути частковим і до того ж в деякому підалфавіті основного алфавіту. В цьому випадку необхідно добавити один новий стан, з якого під дією будь-якого x із X можна перейти лише в себе, і в який із всіх інших станів ведуть переходи під дією будь-якого x із X . Подія, яка представляється в даному автоматі, буде незмінною, а сам автомат стане повністю визначеним над алфавітом X .

Розглянемо деякі наслідки, що випливають з основної теореми теорії скінчених автоматів.

Наслідок 6.1.1. Проблема тотожності двох регулярних подій S і T алгоритмічно розв'язна, тобто існує алгоритм, який для довільної пари регулярних подій S і T з'ясовує, чи буде рівність $S = T$ тотожністю.

Д о в е д е н н я. Синтезуємо автомати для регулярних подій S і T , детермінізуємо їх, а потім застосуємо до них алгоритм перевірки еквівалентності. Якщо автомати еквівалентні, то $S = T$ — тотожність. У протилежному випадку рівність $S = T$ не є тотожністю.

Наслідок 6.1.2. Доповнення регулярної події і перетин двох регулярних подій — регулярні події.

Д о в е д е н н я. Нехай регулярна подія S представлена в автоматі $A(S) = (A, X, f, a_0, F)$, синтезованому за подією S . Тоді доповнення $S' = F(X) \setminus S$ події S , очевидно, представлено в автоматі $A(S') = (A, X, f, a_0, A \setminus F)$. Ясно, що налаштований таким чином X -автомат буде представляти множину слів $F(X) \setminus S$. Якщо тепер провести аналіз побудованого автомата, то можна знайти регулярний вираз для події $S' = F(X) \setminus S$.

Нехай S і T — довільні регулярні події, $A(S) = (A, X, f, A_0, F)$ і $A(T) = (B, X, f', B_0, F')$ — синтезовані за цими подіями автомати. Для того щоб побудувати автомат, який представляє подію $S \cap T$, розглянемо автомат $A(S \cap T) = (A \times B, X, \bar{f}, A_0 \times B_0, F \times F')$, де $A \times B$ — декартовий добуток множин A і B , $F \times F'$ — декартовий добуток множин F і F' , а функція переходів задається рівнянням

$$\bar{f}((a, b), x) = (f(a, x), f(b, x)).$$

Очевидно, що всяке слово, яке належить події $S \cap T$, буде переводити початковий стан автомата $A(S \cap T)$ в один із його заключних станів, а також і автомati $A(S)$ і $A(T)$. Якщо тепер до автомата $A(S \cap T)$ застосувати алгоритм аналізу, то можна одержати регулярний вираз для події $S \cap T$.

Наслідок доведено.

Наслідок 6.1.3. Алгебра регулярних подій в алфавіті X замкнута відносно операцій доповнення перетину і об'єднання і, значить, являє собою булеву алгебру.

Має місце аналогічне твердження і для всіх подій, які можуть бути представлені в заданому X -автоматі.

Наслідок 6.1.4. Множина всіх подій, які можуть бути представлені в заданому X -автоматі, становить булеву алгебру.

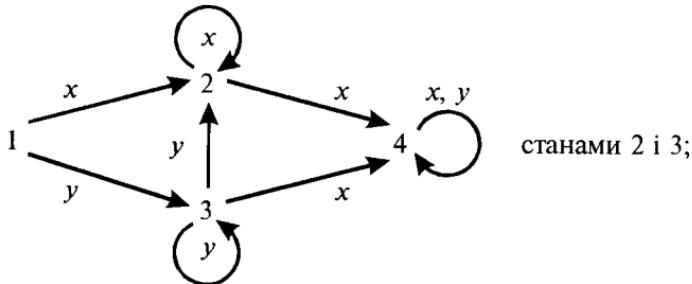
Д о в е д е н н я. Нехай $A = (A, X, f, a_0, F)$ — ініціальний X -автомат. Якщо подія S представлена в автоматі A множиною станів F , а подія T — множиною станів F' , то подія $F(X) \setminus S$ представлена в автоматі A множиною станів $A \setminus F$, подія $S \cup T$ —

множиною станів $F \cup F'$, а подія $S \cap T$ — множиною станів $F \cap F'$. Якщо добавити до множини станів автомата A один стан, який представляє неможливу подію, то звідси випливає справедливість нашого твердження.

Задачі та вправи

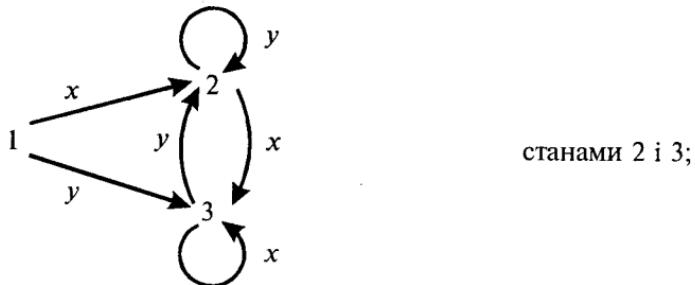
1. Дайте повне доведення твердження 6.1.1.
2. Користуючись алгоритмом аналізу, побудуйте регулярний вираз, представлений в автоматі

a)



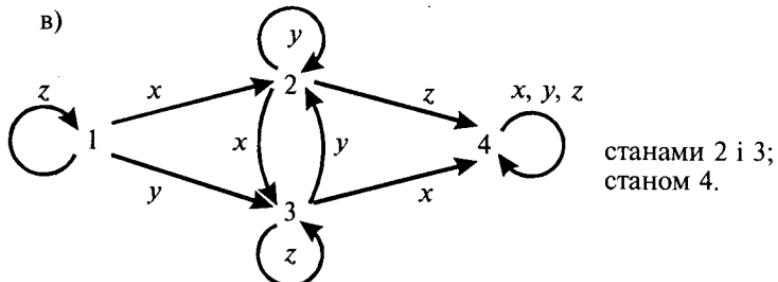
станами 2 і 3;

б)



станами 2 і 3;

в)



станами 2 і 3;
станом 4.

3. Дайте повне обґрунтування правила 2.

4. Дайте повне обґрунтування правила 3.

5. Користуючись алгоритмом синтезу, побудуйте діаграмами переходів автоматів, в яких представлені регулярні події:

- a) $S = x\{y\} \vee y\{x\}$;
- б) $T = y \vee x\{y\}$;
- в) $V = x\{y\} \vee \{xy\} \vee \{x\}y$;
- г) $U = (x \vee y)\{xy\}$.

6. Слова $xy...z$ і $z...yx$ називаються симетричними (див. § 3.3, операція conv). Нехай S — деяка подія, а S' — подія, яка складається із всіх слів, симетричних словам із S . Покажіть, що подія S' теж регулярна.

7. Подія, яка одержується з події S в результаті вилучення деякого символу $x \in X$ із всіх слів події S , називається **x -ануляцією** події S (див. § 3.3, операція sub). Покажіть, що x -ануляція регулярної події S є регулярною подією.

8. Нехай $S = (xy \vee x\{y\})$, $T = (\{x\} \vee \{xy\})$. Побудуйте автомати $A(S)$ і $A(T)$ за даними регулярними виразами. Знайдіть регулярні вирази для подій $(S \cap T)$ і $(S \cup T)$.

§ 6.2. ПРАКТИЧНІ МЕТОДИ АНАЛІЗУ І СИНТЕЗУ СКІНЧЕННИХ АВТОМАТІВ

1. РІВНЯННЯ В АЛГЕБРІ ПОДІЙ

У попередньому параграфі, застосовуючи розроблені алгоритми аналізу і синтезу, ми впевнилися, що ці алгоритми досить громіздкі. Це заважає їх використанню на практиці. У цьому параграфі ми розглянемо зручніші алгоритми аналізу та синтезу скінченних автоматів. Практичний алгоритм аналізу автоматів можна побудувати, розглядаючи рівняння в алгебрі регулярних подій.

Рівнянням в алгебрі подій називається рівність вигляду

$$X = t(X, P, Q, \dots, R),$$

де t — терм алгебри регулярних подій, в якому X — невідома подія, а P, Q, \dots, R — відомі події.

Рівняння називається **лінійним**, якщо жоден одночлен його правої частини не включає X більше одного разу, і це включення не стоїть під знаком ітерації. Загальний вигляд лінійного рівняння такий:

$$X = S_1XR_1 \vee S_2XR_2 \vee \dots \vee S_kXR_k \vee Q, \quad (6.2.1)$$

де S_i, R_i, Q — відомі події. З цього означення випливає, що, наприклад, рівняння $X = R\{X\}Q$, $X = XRX \vee P$ не є лінійними.

Лінійне рівняння вигляду (6.2.1) називається **ліволінійним (праволінійним)**, якщо всі множники, які стоять ліворуч (праворуч) від невідомого, дорівнюють пустому слову e , тобто $S_i = e$ ($R_i = e$), $i = 1, \dots, k$. Очевидно, що всяке лівлінійне (праволінійне) рівняння можна звести до вигляду

$$X = XR \vee Q \quad (X = RX \vee Q). \quad (6.2.2)$$

Розглянемо розв'язки ліволінійного рівняння. Для цього підставимо в праву частину рівняння (6.2.2) вираз $XR \vee Q$ і одержимо нове рівняння

$$X = (XR \vee Q)R \vee Q = XR^2 \vee QR \vee Q.$$

Виконавши такі підстановки кілька разів, прийдемо до рівняння

$$X = XR^n \vee QR^{n-1} \vee \dots \vee QR \vee Q = XR^n \vee Q(R^{n-1} \vee \dots \vee R \vee e).$$

Звідси випливає, що подія $Q(R^{n-1} \vee \dots \vee R \vee e)$ входить до розв'язку даного рівняння. Оскільки це властиво для всякого натурального n , то звідси маємо, що подія $Q\{R\}$ повинна належати до шуканого значення невідомого X . Але, підставивши в (6.2.2) замість X значення $Q\{R\}$, маємо $Q\{R\} = Q\{R\}R \vee Q = Q\{R\}R \vee e = Q\{R\}$, а отже, $X = Q\{R\}$ є розв'язком рівняння (6.2.2).

Теорема 6.2.1. Якщо $e \notin R$, то рівняння $X = XR \vee Q$ має єдиний розв'язок $X = Q\{R\}$.

Д о в е д е н н я. Нехай X' — розв'язок рівняння (6.2.2) і p — довільне слово з X' . Покажемо індукцією за довжиною слова p , що воно буде належати $Q\{R\}$. Це підтверджує, що $X \subseteq Q\{R\}$, а оскільки раніше було показано, що $Q\{R\}$ входить у всякий розв'язок, то це і буде означати, що подія $Q\{R\}$ — єдиний розв'язок нашого рівняння.

Якщо $p = e$, то e повинно входити в $XR \vee Q$, але оскільки подія XR не може включати цього слова, то $e \in Q$ і, значить, $p \in Q\{R\}$.

Припустимо тепер, що для всіх слів, які належать розв'язку X' і мають довжину меншу ніж p , ми показали, що вони належать також і $Q\{R\}$. Із того, що $p \in X'$, випливає, що $p \in X'R$ або $p \in Q$. В останньому випадку включення $p \in Q\{R\}$ очевидне. Припустимо, що $p \in X'R$. У цьому випадку слово p можна подати у вигляді $p = p'p''$, де $p' \in X'$, $p'' \in Q$, і довжина слова p'' не є нульовою. Але це означає, що довжина слова p' менша довжини слова p і за припущенням індукції $p' \in Q\{R\}$, а тоді $p = p'p''$ також належить $Q\{R\}$, що і доводить теорему.

Наслідок 6.2.1. Якщо $e \notin R$, R і Q — регулярні події, то розв'язок рівняння $X = XR \vee Q$ — теж регулярна подія.

Аналогічно теоремі 6.2.1 можна довести таке твердження.

Теорема 6.2.2. Якщо $e \in R$, то всякий розв'язок рівняння $X = XR \vee Q$ має вигляд $(Q \vee T)\{R\}$, де T — довільна подія.

2. СИСТЕМИ ЛІНІЙНИХ РІВНЯНЬ

Розглянемо тепер систему лінійних лівосторонніх рівнянь:

$$\left\{ \begin{array}{l} X_1 = X_2 S_{11} \vee X_2 S_{21} \vee \dots \vee X_n S_{n1} \vee R_1 \\ X_2 = X_2 S_{12} \vee X_2 S_{22} \vee \dots \vee X_n S_{n2} \vee R_2 \\ \dots \\ X_n = X_2 S_{1n} \vee X_2 S_{2n} \vee \dots \vee X_n S_{nn} \vee R_n. \end{array} \right. \quad (6.2.3)$$

Ця система має єдиний розв'язок, коли $e \in S_{ij}$ ($i, j = 1, 2, \dots, n$). Розглянемо метод розв'язання такої системи, який називається *методом послідовного виключення невідомих*. Суть цього методу добре відома з лінійної алгебри під назвою методу Гаусса і полягає в тому, щоб з першого (не обов'язково по порядку) рівняння виразити невідоме X_1 через інші невідомі. Знайшовши вираз для X_1 , підставляємо його в решту рівнянь і одержуємо вираз для X_2 і так далі доти, доки не буде знайдено вираз для X_n , який вже не містить невідомих. Після цього внаслідок зворотного процесу знаходимо значення для $X_{n-1}, X_{n-2}, \dots, X_2, X_1$, виражені через події.

Теорема 6.2.3. Якщо всі коефіцієнти S_{ij} системи (6.2.3) не включають пустого слова e , то ця система має єдиний розв'язок, який може бути знайдений методом послідовного виключення невідомих. Цей розв'язок буде регулярним, якщо всі S_{ij} і R_i ($i, j = 1, \dots, n$) регулярні.

Доведення виконується індукцією за числом рівнянь у системі. Якщо система складається з одного рівняння, то твердження теореми випливає з теореми 6.2.1. Нехай теорема справедлива для всіх систем з числом рівнянь до $n - 1$ включно. Покажемо, що вона справедлива для системи, яка складається з n рівнянь.

Для цього з останнього рівняння знайдемо вираз для X_n відносно решти невідомих. За теоремою 6.2.1 маємо

$$X_n = (X_1 S_{1n} \vee X_2 S_{2n} \vee \dots \vee X_{n-1} S_{n-1n} \vee R_n) \{S_{nn}\}. \quad (6.2.4)$$

Підставимо одержаний вираз для X_n в перші $n - 1$ рівняння системи (6.2.3). Приходимо до системи

$$\left\{ \begin{array}{l} X_1 = X_1(S_{11} \vee S_{1n} \{S_{nn}\} S_{n1}) \vee \dots \vee \\ \quad \vee X_{n-1}(S_{n-11} \vee S_{n-1n} \{S_{nn}\} S_{n1}) \vee (R_1 \vee R_n \{S_{nn}\} S_{n1}) \\ \dots \\ X_{n-1} = X_1(S_{n-11} \vee S_{n-1n} \{S_{nn}\} S_{nn-1}) \vee \dots \vee \\ \quad \vee X_{n-1}(S_{n-1n-1} \vee S_{n-1n} \{S_{nn}\} S_{n1}) \vee (R_{n-1} \vee S_n \{S_{nn}\} S_{nn-1}). \end{array} \right. \quad (6.2.5)$$

Неважко переконатися, що одержана система задовільняє умови теореми, а тоді, за припущенням індукції, її єдиний розв'язок може бути знайдений методом послідовного вилучення невідомих. Нехай $\bar{X}_1, \bar{X}_2, \dots, \bar{X}_{n-1}$ є розв'язком системи (6.2.5). Підставимо їх в систему (6.2.4). Одержано сукупність подій $\bar{X}_1, \bar{X}_2, \dots, \bar{X}_{n-1}, (\bar{X}_1 S_{1n} \vee \dots \vee \bar{X}_{n-1} S_{(n-1)n} \vee R_n) \{S_{nn}\}$, які є розв'язком системи (6.2.3) і будуть регулярними, якщо такими є події $\bar{X}_1, \bar{X}_2, \dots, \bar{X}_{n-1}$ і коефіцієнти S_{in} і R_n ($i = 1, 2, \dots, n$).

Теорема доведена.

Зауважимо, що все сказане про лінійні лівосторонні рівняння і системи справедливе і для лінійних правосторонніх рівнянь і систем. Слід також зазначити, що дані результати не стосуються систем, в яких одночасно зустрічаються як лівосторонні, так і правосторонні рівняння. Більше того, може трапитися так, що розв'язок такої системи може бути нерегулярним, незважаючи на регулярність всіх коефіцієнтів системи.

3. ЗАСТОСУВАННЯ РІВНЯНЬ В АЛГЕБРІ ПОДІЙ ДО ЗАДАЧ АНАЛІЗУ І СИНТЕЗУ СКІНЧЕННИХ АВТОМАТІВ

Нехай $A = (F, X, f, a_1, F)$ — настроєний автомат, де $A = \{a_1, a_2, \dots, a_n\}$. Для кожного стану a_i ($i = 1, \dots, n$) символом S_i позначимо подію

$$\{p \mid f(a_i, p) = a_i\}.$$

Задачу аналізу автомата можна, очевидно, сформулювати як задачу пошуку регулярного виразу для диз'юнкції тих подій S_i , індексами яких позначаються заключні стани з F . Для знаходження розв'язку цієї задачі складемо систему рівнянь відносно S_1, S_2, \dots, S_n :

$$S_i = \bigvee_{\substack{j=1 \\ f(a_j, x) = a_i}}^n S_j x \vee S_{1i}, \quad (6.2.6)$$

де

$$S_{1i} = \begin{cases} e, & \text{якщо } i = 1 \\ \emptyset, & \text{якщо } i \neq 1 \end{cases}, \quad i = 1, \dots, n.$$

Коефіцієнти при невідомих S_i в цій системі, очевидно, не містять пустого слова. Отже, система має єдиний розв'язок, який може бути знайдений методом послідовного вилучення невідомих.

Теорема 6.2.4. Нехай $A = (A, X, f, a_1, F)$ — ініціальний автомат з множиною станів $\{a_1, a_2, \dots, a_n\}$. Тоді сукупність подій $S_i = \{p \mid f(a_i, p) = a_j\} (i = 1, \dots, n)$ є розв'язком системи (6.2.6).

Д о в е д е н н я. Внаслідок того, що сукупність подій S_i канонічна, а розв'язок системи (6.2.6) єдиний, достатньо показати, що всяке слово q із $F(X)$, яке належить події S_i , також належить i -му компоненту розв'язку системи (6.2.6) і тільки йому. Доведено це індукцією за довжиною слова q .

Для пустого слова — єдиного слова довжини нуль — це має місце згідно з означенням події S_j і тому, що всі коефіцієнти при невідомих не містять пустого слова.

Припустимо тепер, що наша теорема справедлива для всіх слів довжини, меншої, ніж довжина слова q , і що $q = q'x \in \{p \mid f(a_i, x) = a_j\}$, а $q' \in \{q \mid f(a_1, q') = a_j\}$. З цих включень випливає, що $f(a_j, x) = a_i$, а тоді за припущенням індукції (q' належить j -му компоненту і тільки йому) і за побудовою системи рівнянь (данок Sx входить тільки в i -те рівняння) маємо, що q належить i -му компоненту і тільки йому.

Приклад 6.2.1

Виконати аналіз автомата (рис. 6.2.1) за допомогою апарату системи рівнянь в алгебрі регулярних подій, в якому $a_0 = 1$, а $F = \{2, 3\}$.

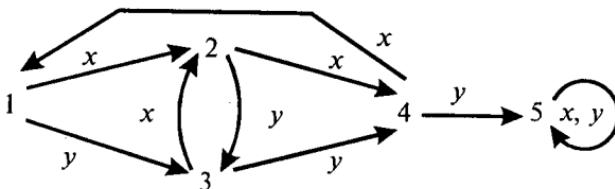


Рис. 6.2.1

З діаграми переходів автомата знаходимо систему

$$\begin{aligned} S_1 &= S_4x \vee e; \\ S_2 &= S_1x \vee S_3x; \\ S_3 &= S_1y \vee S_2y; \\ S_4 &= S_2x \vee S_3y; \\ S_5 &= S_4y \vee S_5(x \vee y). \end{aligned}$$

Необхідно знайти диз'юнкцію другого і третього компонентів розв'язку системи. Ці компоненти залежать від другого і четвертого і жоден із них не залежить від п'ятого компонента. Тому ми можемо одразу обмежитися лише першими чотирма рівняннями. Оскільки розв'язок системи єдиний і не залежить від по-

рядку вилучення невідомих, то дозволимо собі вибрати цей порядок.

Підставимо друге рівняння в третє і розв'яжемо його відносно S_3 . Одержано $S_3 = S_1 (y \vee xy)\{xy\}$.

Підставимо тепер третє рівняння в друге і розв'яжемо його відносно S_2 . Одержано $S_2 = S_1 (x \vee yx)\{yx\}$.

Підставимо одержані вирази для S_3 і S_2 в четверте рівняння:

$$S_4 = S_1 (x \vee yx)\{yx\}x \vee S_1 (y \vee xy)\{xy\}y.$$

Цей вираз для S_4 підставимо в перше рівняння і розв'яжемо його відносно S_1 :

$$S_1 = ((x \vee yx)\{yx\}xx \vee (y \vee xy)\{xy\}y\{x\}).$$

Шукають подією є диз'юнкція $S_2 \vee S_3$, яка виражається через S_1 таким чином:

$$S_2 \vee S_3 = S_1 ((y \vee xy)\{xy\} \vee (x \vee yx)\{yx\}).$$

Остаточно, підставивши вираз для S_1 в цю диз'юнкцію, маємо

$$S_2 \vee S_3 = ((x \vee yx)\{yx\}xx \vee (y \vee xy)\{xy\}y\{x\}) ((y \vee xy)\{xy\} \vee (x \vee yx)\{yx\}).$$

Отже, аналіз автомата методом складання і розв'язку рівнянь в алгебрі подій досить зручний, оскільки він використовує добре відомі поняття лінійної алгебри. Слід також зауважити, що при цьому одержується досить компактний запис результату без застосування тотожних перетворень, в той час як у попередньому методі ці перетворення застосовувались неодноразово. ▲

4. ОСНОВНИЙ АЛГОРИТМ СИНТЕЗУ СКІНЧЕННИХ АВТОМАТІВ

Наведений нижче алгоритм синтезу скінченних автоматів базується на понятті входження букв у регулярний вираз. Даний алгоритм має ті переваги перед відповідним алгоритмом із § 5.1, що він досить практичний у користуванні і може оперувати не лише одним регулярним виразом, а й будь-якою скінченою сукупністю таких виразів.

Входженням букви x в регулярний вираз R є трійка $(R'xR')$, де R' і R'' — частини запису терма R , такі, що $R'xR''$ є записом терма R . Запис R' називають **префіксом входження**, а запис R'' — його **суфіксом**. Зауважимо, що префікс і суфікс входження взагалі можуть не бути записами регулярних виразів, наприклад через порушення балансу дужок.

Поняття входження букви у вираз можна узагальнити на підвирази: входженням підвиразу Q у вираз R будемо називати трійку $(R'QR'')$, де R' — префікс входження, R'' — суфікс входження, $R'QR''$ — запис виразу R . Наприклад, вираз $x(xy \vee ux)\{xy\}$ включає два входження підвиразу xy : одне з них — $(x, xy, \vee ux)\{xy\}$, а друге — $(x(xy \vee ux), \{xy\})$. При фіксованому входженні виразу Q у вираз R входження букв однозначно індукують входження цих же букв у більш широкий підвираз виразу R . Входження $(R'xR'')$ букв x у вираз R індукується входженням (R_1QR_2) підвиразу Q у вираз R і входженням $(R_1'xR_2')$ у цей підвираз, якщо $R' = R_1R_1'$ і $R'' = R_2'R_2$.

Нехай R — регулярний вираз, і X — множина його регулярних підвиразів. Розглянемо множину всіх входжень букв алфавіту X у вирази із R , ототожнюючи при цьому різні входження однієї і тієї ж букви, якщо одне з них індукується другим. На множині входжень індуктивно визначимо початкові і заключні входження, а також відношення слідування. Для скорочення записів умовимося ототожнювати там, де це не викликає непорозумінь, входження букви з самою буквою.

Нехай $L(S)$ — множина початкових входжень букв у вираз S . Ця множина визначається за такими правилами:

- 1) якщо $S = e$ або $S = \emptyset$, то $L(S) = \emptyset$, і якщо S — елементарна подія ($S = \{x\}$), то $L(S) = \{x\}$;
- 2) якщо $S = S' \vee S''$, то $L(S) = L(S') \cup L(S'')$;
- 3) якщо $S = S'S''$, то $L(S) = L(S')$, якщо $e \notin S'$, і $L(S) = L(S') \cup L(S'')$ в протилежному випадку;
- 4) якщо $S = \{S'\}$, то $L(S) = L(S')$.

Аналогічно індукцією за числом операцій у виразі S визначимо і множину $R(S)$ заключних входжень букв у вираз S :

- 1) якщо $S = e$ або $S = \emptyset$, то $R(S) = \emptyset$, і якщо S — елементарна подія ($S = \{x\}$), то $R(S) = \{x\}$;
- 2) якщо $S = S' \vee S''$, то $R(S) = R(S') \cup R(S'')$;
- 3) якщо $S = S'S''$, то $R(S) = R(S'')$, якщо $e \notin S''$, і $R(S) = R(S') \cup R(S'')$ в протилежному випадку;
- 4) якщо $S = \{S'\}$, то $R(S) = R(S')$.

Перейдемо тепер до поняття слідування на множині входжень букв одного і того ж виразу.

Нехай x і y — деякі фіксовані входження відповідних букв у вираз S . Позначення $x \vdash y (S)$ означає, що входження букви y слідує за входженням букви x у вираз S . Дамо індуктивне визначення відношення слідування:

- 1) якщо $S = S' \vee S''$, то $x \vdash y (S) \Leftrightarrow x \vdash y (S')$ або $x \vdash y (S'')$;

2) якщо $S = S'S''$, то $x \mid y (S) \Leftrightarrow x \mid y (S')$ або $x \mid y (S'')$ або $x \in R(S')$ і $y \in L(S'')$;

3) якщо $S = \{S'\}$, то $x \mid y (S) \Leftrightarrow x \mid y (S')$ або $x \in R(S')$ і $y \in L(S')$.

Теорема 6.2.5. Якщо S — регулярний вираз, то слово $x_1x_2\dots x_n$ належить події, яка представлена цим виразом, тоді і тільки тоді, коли існує послідовність $z_1z_2\dots z_n$ входжень букв у вираз S , така, що:

1) $z_i \in$ входженням букв x_i ($i = 1, 2, \dots, n$);

2) $z_i \mid z_{i+1}$, якщо $1 \leq i \leq n - 1$;

3) $z_1 \in L(S)$ і $z_n \in R(S)$.

Доведення проводиться індукцією за числом операцій у виразі S . Твердження очевидне, якщо $S \in \emptyset$, e або елементарною подією.

Нехай воно виконується для виразів S' і S'' . Покажемо його справедливість для виразів $S' \vee S''$, $S'S''$ і $\{S'\}$.

Якщо слово p належить події, яка представлена виразом $S' \vee S''$, то це може бути лише тоді, коли це слово належить події, яка представлена виразом S' або S'' . Але за припущенням індукції і означенням множин $L(S' \vee S'')$ і $R(S' \vee S'')$ для цього слова будуть виконуватися умови 1)—3) теореми.

Якщо $p \in S'S''$, то це включення може бути виконане одним із трьох способів: $p \in S'$ і $e \in S''$; $e \in S'$ і $p \in S''$; $p = p'p''$, $p' \in S'$ і $p'' \in S''$. За припущенням індукції й індуктивних означенень множин початкових і заключних входжень, а також відношення слідування на множині входжень у кожному з трьох випадків, коли виконується включення $p \in S'S''$, будуть виконуватися умови 1)—3).

Якщо ж $p \in \{S'\}$, то це означає, що $p = p_1p_2\dots p_n$ ($n \geq 1$) і $p_i \in S'$. Але звідси за припущенням індукції, означенням множин $L(\{S'\})$, $R(\{S'\})$ і відношенням слідування випливає справедливість умов 1)—3).

Теорема доведена.

Теорема 6.2.6. Існує алгоритм, який за будь-якою скінченою системою регулярних виразів в алфавіті X буде скінчений налаштований X -автомат, що представляє кожний із регулярних виразів даної системи деякою множиною своїх станів. Число станів у цьому автомата не перевищує $2^n + 1$, де n — сумарне число різних входжень букв алфавіту X в дані вирази.

Доведення. Станами автомата A , який необхідно синтезувати, за винятком початкового стану a_0 , будуть служити множини входжень букв у дану систему регулярних виразів. При цьому не виключається і використання пустої множини входжень.

Функції переходів $f(a, x)$ автомата A дамо за допомогою таких двох правил:

а) для довільного $x \in X$ стан $f(a_0, x)$ являє собою множину всіх початкових входжень букв x в усі вирази даної сукупності виразів;

б) якщо стан a представляється деякою множиною входжень, то для всякого $x \in X$ стан $f(a, x)$ є множиною всіх входжень букв x , яке слідує за входженнями букв, що представляють стан a .

Зауважимо, що станами автомата A є не довільні підмножини входження букв, а тільки ті, які досяжні з початкового стану під дією якого-небудь слова відповідно до правил, що задають функцію переходів.

З теореми 6.2.5 і способу побудови переходів автомата A видно, що непусте слово p належить деякій події, представлений виразом S у початковій сукупності виразів, тоді і тільки тоді, коли під дією цього слова автомат A переходить із початкового стану в стан, який розглядається як множина входжень букв і включає хоча б одне заключне входження для виразу S . За побудовою автомата A рівність $f(a_0, p) = a_0$ можлива тільки тоді, коли слово p пусте і, значить, подія, яка представлена станом a_0 , — пусте слово.

З усього сказаного випливає, що подія, яка відповідає регулярному виразу S , представлена в автоматі A множиною станів, які містять заключні входження букв у вираз S , і якщо ця подія включає пусте слово, то до цієї множини станів слід віднести початковий стан a_0 .

Очевидно, що число станів побудованого таким чином автомата не перевищує $2^n + 1$.

Підводячи підсумки цієї теореми, сформулюємо в явному вигляді правила синтезу автомата, за якими можна реалізувати алгоритм синтезу автомата.

Правило 1. Розмітити входження букв у початкових регулярних виразах.

Правило 2. Стан, який позначається a_0 , об'явити початковим і поки що єдиним станом автомата.

Правило 3. У відповідності з правилом а) (означення функції переходів) заповнити стовпчик стану a_0 в таблиці переходів так, що у цьому разі об'являються станами автомата всі стани, які є початковими входженнями. Відвести новим станам відповідні стовпчики в таблиці переходів.

Правило 4. Якщо в таблиці переходів усі стовпчики заповнені, то перейти до виконання правила 6, в протилежному випадку вибрати незаповнений стовпчик і перейти до правила 5.

Правило 5. Відповідно з правилом б) (означення функції переходів) заповнити вибраний стовпчик множиною входжень букв, об'явити при цьому множини входжень, які раніше не зустрічалися, станами автомата і поставити у відповідність кожному з них новий стовпчик у таблиці переходів.

Правило 6. Для кожного регулярного виразу R із початкової сукупності виразів об'явити заключними станами всі ті стани, які містять хоча б по одному заключному входженю букви у вираз R . При цьому якщо представлений вираз R включає пусте слово, то початковий стан теж об'явити заключним для цього виразу.

Для застосування цього алгоритму на практиці необхідно вибрати зручний спосіб задання різних входжень букв у вирази з даної сукупності. Для цього введемо нумерацію входжень букв цілими числами, при якій кожне входження одержить свій особливий, тільки йому притаманний номер (порядок нумерації довільний). При нумерації входжень букв їх номери будемо ставити біля відповідних букв (правіше і нижче) і називати *індексами входжень*. Процес нумерації входжень букв у вирази будемо називати *розміченням*.

Застосовуючи алгоритм синтезу до заданої розміченої сукупності виразів і ототожнюючи входження букв з їх індексами, будемо представляти стани множинами індексів.

Приклад 6.2.2

Побудувати автомат A , який представляє регулярні вирази в алфавіті $\{x, y\}$, що мають вигляд

$$\begin{aligned}S_1 &= (x \vee e)\{yx\} \vee xy, \\S_2 &= \{yx\}(x \vee yx).\end{aligned}$$

Розв'язок задачі почнемо з розмітки виразів S_1 і S_2 :

$$S_1 = (x_1 \vee e)\{y_2x_3\} \vee x_4y_5, \quad S_2 = \{y_6x_7\}(x_8 \vee y_9x_{10}).$$

Множинами початкових і заключних входжень букв у вирази відповідно до їх означення будуть множини $\{1, 2, 4, 6, 8, 9\}$ і $\{1, 3, 5, 8, 10\}$. Відношення слідування на множині входжень задамо парами $(1, 2), (2, 3), (3, 2), (4, 5), (6, 7), (7, 6), (7, 8), (7, 9)$ і $(9, 10)$, де друге число означає входження, яке слідує за входженням, якому відповідає перше число.

Виконуючи процес породження станів автомата і заповнюючи таблицю переходів (правила 2—5), приходимо до такої множини станів:

$a_0, a_1 = \{1, 4, 8\}, a_2 = \{2, 6, 9\}, a_3 = \{2, 5\}, a_4 = \{3, 7, 10\}, a_5 = \{3\}, a_6 = \{8\}, a_7 = \{2\}$ і $a_8 = \{\emptyset\}$ (цей стан у таблиці переходів

позначений символом *). Остаточно одержимо таблицю переходів.

Вхідні сигнали	Стани автомата								*
	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	
x	a_1	*	a_4	a_5	a_6	*	*	a_5	*
y	a_2	a_3	*	*	a_2	a_7	*	*	*

При подальшій роботі алгоритму синтезу (правило 6) знаходимо множини станів, які представляють ті ж події, що й вирази S_1 і S_2 . Цими множинами будуть: для виразу S_1 — множина $\{a_0, a_1, a_3, a_4, a_5\}$ (зауважимо, що пусте слово належить виразу S_1), для виразу S_2 — множина $\{a_1, a_4, a_5\}$.

Контрольні питання

1. Яке рівняння в алгебрі подій називається лінійним?
2. Що являє собою розв'язок системи лінійних рівнянь в алгебрі подій?
3. Чи всякий автомат можна синтезувати за заданим регулярним виразом?
4. Чи для всякого скінченного автомата можна побудувати представлений у ньому регулярний вираз?
5. Які ви знаете правила синтезу автомата? Назвіть їх.

Задачі та вправи

1. Застосовуючи алгоритм аналізу до автоматів з вправи 1 § 6.1, побудуйте регулярні вирази для відповідних автоматів.
2. Побудуйте регулярний вираз для автомата, який представляє напівгрупу слів, що породжена такими словами: $\{ab, abc, ba, cba\}$.
3. Використовуючи алгоритм синтезу скінчених автоматів, побудуйте автомат, який представляє такі регулярні вирази:

$$\begin{aligned}S_1 &= (x \vee e)\{xy\}, \\S_2 &= \{yx\}(x \vee yx), \\S_3 &= (x \vee e)\{x\} \vee y, \\S_4 &= \{y\}(x \vee y).\end{aligned}$$

4. Користуючись методом послідовного виключення невідомих, знайдіть розв'язки системи рівнянь:

$$\begin{aligned}X &= XR \vee YQ \vee Z, \\Y &= XP \vee YA \vee B.\end{aligned}$$

§ 6.3. МІНІМІЗАЦІЯ СКІНЧЕННИХ АВТОМАТІВ БЕЗ ВИХОДІВ

У попередньому розділі (див. § 6.2) ми розглянули загальний метод побудови приведеного автомата для заданого. Цей метод базувався на застосуванні операції розщеплення класів і у випадку скінчених автоматів давав алгоритм побудови приведеного автомата. Розглянемо детальніше процес мінімізації X -автоматів.

Наземо пару (K, x) елементарним дільником або елементарним подрібнювачем, де K — клас деякого розбиття P , x — вхідний символ автомата A , а множину всіх дільників позначимо декартовим добутком $P \times X$. Не будемо описувати типів змінних, які зустрічаються в цьому алгоритмі, умовимося лише позначати класи станів автомата буквами K, L, L', L'', P_0 , а множини класів (розбиття) — P, P', P_0 . Початкове значення змінної P_0 дорівнює початковому розбиттю множини станів автомата A на 1-класи чи 0-класи.

Алгоритм MA1 (A)

початок.

$$P' := P_0;$$

ділити: $P := P'$;

$$P' := P' / P' \times X;$$

Якщо $P \neq P'$, то перейти на ділити, інакше зупинитись;
кінець.

1. ЗАГАЛЬНИЙ АЛГОРИТМ МІНІМІЗАЦІЇ

Застосовуючи трансформаційний метод побудови алгоритмів, за алгоритмом MA1 можна побудувати досить ефективний алгоритм мінімізації скінчених автоматів без виходу. Суть трансформаційного методу продемонструємо нижче при побудові алгоритму мінімізації для ациклічних автоматів, а зараз нагадаємо деякі означення теорії скінчених автоматів без виходу.

Поширимо, як і раніше, функцію переходів f на всю напівгрупу слів $F(X)$, покладаючи для всякого $p = xp'$ ($x \in X, p' \in F(X)$) і $a \in A$

$$f(a, p) = f(f(a, x), p').$$

При цьому якщо функція $f(a, x)$ не визначена, то і $f(a, p)$ теж не визначена. Якщо ж $f(a, p)$ визначена і дорівнює b , то говорять, що слово p переводить автомат A із стану a в стан b .

Відношення еквівалентності (\sim) на множині станів автомата без виходів визначається таким чином:

$$a \sim a' \Leftrightarrow (\forall p \in F(X)) (f(a, p) \in F \Leftrightarrow f(a', p) \in F).$$

Відношення \sim , як легко переконатися, є відношенням конгруентності на множині A відносно функції переходів f , значить, можна говорити про фактор-автомат B / \sim . Автомат, в якого всі стани попарно нееквівалентні, називається **приведеним**. Приведеним автомatem для заданого автомата B якраз і служить фактор-автомат B / \sim .

Побудова приведеного автомата для заданого разом із задачами його аналізу та синтезу — це одна з основних задач теорії скінчених автоматів. Крім того, з попереднього розділу випливає, що, виконуючи мінімізацію автоматів без виходу, ми не обмежуємо загальності нашого розгляду.

Як ми вже знаємо, побудова приведеного автомата — ітераційний процес, при якому будуються максимальні класи еквівалентності. Побудова починається з деякого апріорного розбиття, наприклад розбиття на два класи — заключні стани і всі останні стани початкового автомата, і продовжується доти, доки одержані класи уже не подрібнюються на дрібніші. Оскільки автомат скінчений, то цей процес завжди закінчується через скінченне число ітерацій. Подрібнення виконується таким чином, що для всякого поточного розбиття стани з різних класів між собою нееквівалентні. Перейдемо тепер до означення функції подрібнення класів і виявлення деяких її властивостей.

Нехай K, L, L' — деякі підмножини станів автомата $A = (A, X, f, F)$ і x — деякий символ із X . Означення функції подрібнення ($/$) можна дати таке:

$$L / (K, x) = \{L', L''\},$$

де $L' = \{a \in L \mid f(a, x) \in K\}$, $L'' = L \setminus L'$, якщо $L' \neq \emptyset$ і $L'' \neq \emptyset$, інакше — $L / (K, x) = L$.

Безпосередньо з означення випливають такі співвідношення для функції $/$:

- 1) $(L / (K, x)) / (K, x) = L / (K, x);$
- 2) $(L / (K, x)) / (K', x') = (L / (K', x')) / (K, x);$
- 3) $(L / (K', x)) / (K'', x) = (L / (K, x)) / (K', x) =$
 $= (L / (K, x)) / (K'', x) = (((L / (K, x)) / (K', x)) / (K'', x)),$

де $K = K' \cup K'', K' \cap K'' = \emptyset$.

На основі цих властивостей в [18, 21] виконано побудову ефективного алгоритму, виходячи з наведеного вище алгоритму MA1.

Алгоритм мінімізації скінчених автоматів без виходу, який одержано трансформаційним методом з алгоритму MA1, має та-кий вигляд.

2. АЛГОРИТМ МІНІМІЗАЦІЇ СКІНЧЕННИХ АВТОМАТІВ БЕЗ ВИХОДІВ

Алгоритм MA (A)

початок.

$R := R_0;$

ДРОБ := $R_0 \times X;$

Поки ДРОБ $\neq \emptyset$ виконувати

 Взяти (K, x) із ДРОБ;

 Для всіх L із R виконати

$L' := \{a \in L \mid f(a, x) \in K\};$

$L'' := \{a \in L \mid f(a, x) \notin K\};$

 якщо $L' \neq \emptyset$ і $L'' \neq \emptyset$ то

 Замінити L в R на $\{L', L''\}$;

 Для всіх x із X виконати

 якщо (L, x) є ДРОБ то Замінити (L, x) в ДРОБ на $\{(L', x), (L'', x)\}$;

 інакше ДРОБ := ДРОБ $\cup G((L', x), (L'', x))$;

 кя;

 кц;

 кя;

 кц;

кінець.

Конструкції, які використовуються в наведеному алгоритмі, близькі до звичайних конструкцій, якими користуються в програмуванні:

- змінні R і ДРОБ мають тип “множина множин”;
- змінні K, L, L', L'', X — “множина”, а значенням функції $G((L', x), (L'', x))$ є та із двох пар, подрібнення по якій простіше.

Семантика оператора “Взяти (K, x) із ДРОБ” така, що після його виконання елемент (K, x) вилучається з множини ДРОБ.

Оператори циклу вигляду “Для всіх ... із ... виконати” діють таким чином, що кожний елемент вказаної в заголовку множини (в даному випадку X або R) вибирається як параметр один раз, порядок вибору несуттєвий, і закінчення циклу настає тоді, коли вся множина перебрана.

В результаті роботи алгоритму MA одержуємо автомат, основна властивість якого виражається таким твердженням.

Теорема 6.3.1. *Фактор-автомат A / \sim є приведеним автоматом для автомата A.*

Д о в е д е н н я суттєво не відрізняється від доведення теореми 4.1.5 і пропонується як вправа.

Приклад 6.3.3

Побудувати приведений автомат для автомата $A = (\{1, 2, 3, 4, 5, 6, 7\}, \{0, 1\}, f, \{7\})$, функція переходів якого задана за допомогою графа рис. 6.3.1.

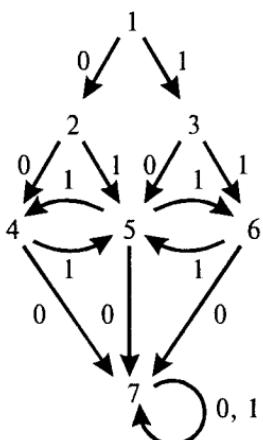


Рис. 6.3.1

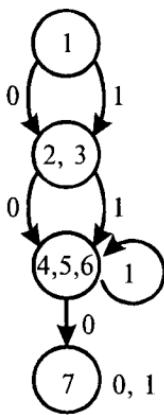


Рис. 6.3.2

Застосовуючи алгоритм МА, послідовно одержуємо: $R_0 = \{(7), (1, 2, 3, 4, 5, 6)\}$, ДРОБ = $\{(7, 0), (7, 1), ((1, 2, 3, 4, 5, 6), 0), ((1, 2, 3, 4, 5, 6), 1)\}$. Подрібнюємо класи по парі $(7, 0)$. Клас (7) при цьому не змінюється, а $(1, 2, 3, 4, 5, 6)$ розбивається на два підкласи: $(1, 2, 3)$ і $(4, 5, 6)$. Після цього ДРОБ = $\{(7, 1), ((1, 2, 3), 0), ((1, 2, 3), 1), ((4, 5, 6), 0), ((4, 5, 6), 1)\}$, а $R = \{(7), (1, 2, 3), (4, 5, 6)\}$.

Подрібнення по парі $(7, 1)$ нічого не змінює. Далі подрібнюємо

$$\begin{aligned} \{4, 5, 6\} / ((4, 5, 6), 1) &= \{4, 5, 6\}; \\ \{1, 2, 3\} / ((4, 5, 6), 1) &= \{\{2, 3\}, \{1\}\}; \\ \{2, 3\} / ((4, 5, 6), 0) &= \{2, 3\}. \end{aligned}$$

Після цього класи більше не змінюються, значить, одержимо автомат рис. 6.3.2. Отже, $\sim = \{(1), (2, 3), (4, 5, 6), (7)\}$. Пере-нумеруємо знову класи і приймемо їх за нові стани. Одержано фактор-автомат, зображенний на рис. 6.3.3.

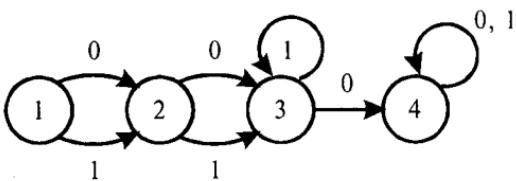


Рис. 6.3.3.

3. АЛГОРИТМ МІНІМІЗАЦІЇ АЦІКЛІЧНИХ АВТОМАТІВ

Тепер розглянемо задачу мінімізації ациклічних скінченних автоматів. Зауважимо, що алгоритм приведення скінченних автоматів, наведений вище, можна застосовувати і до ациклічних автоматів. Але якщо автомат ациклічний, то цей алгоритм можна зробити більш ефективним щодо витрат як часу, так і пам'яті.

Нехай автомат $A = (A, X, f, F)$ ациклічний. Ациклічність автомата A дозволяє розбити множину його станів на рівні так, що

стани верхнього рівня недосяжні із станів нижнього рівня. Формально це записується так.

Нехай $L < L'$ означає, що клас L досяжний з класу L' ; тоді

$$L < L' \Rightarrow L / (L', x) = L. \quad (6.3.1)$$

Очевидно, що відношення $<$ є відношенням часткового порядку. При цьому класи одного рівня не порівнянні між собою.

Розглянемо процес перетворень (трансформацій) алгоритму МА для випадку ациклічних автоматів.

Початковими даними для цих перетворень є алгоритм $MA(A)$, де A — ациклічний автомат, приведення якого необхідно виконати, і співвідношення, яке характеризує ациклічні автомати:

$$(\forall a, a' \in A)(\forall p \in F(X)) (f(a, p) = a' \Rightarrow a \sim a'). \quad (6.3.2)$$

В алгоритмі МА початковим було розбиття множини станів автомата на два класи: $\{F, A \setminus F\}$ — заключних і решти станів. Для випадку ациклічних автоматів за початкове розбиття можна взяти розбиття $R_{\text{rang}} = \{S_0, S_1, \dots, S_m\}$, де

$$\begin{aligned} S_0 &= \{a \in A \mid (\forall x \in X) f(a, x) = w\}; \\ S_{i+1} &= \{a \in A \mid ((\forall x \in X) f(a, x) = w) \vee \\ &\vee f(a, x) \in \bigcup_{j=1}^i S_j\} \& (\exists x \in X) (f(a, x) \in S_i)\}, \end{aligned}$$

w — символ невизначеності. Нееквівалентність станів, які лежать у різних класах, є безпосереднім наслідком співвідношення (6.3.2).

Використовуючи розбиття R_{rang} , перші два оператори алгоритму МА замінимо операторами “ $R := R_{\text{rang}}$ ” і “ДРОБ := $R \times X$.”.

Частковий порядок, властивий множині R_{rang} , дозволяє говорити про частковий порядок на множині R . При цьому класи, які належать до одного рівня, не порівнянні між собою відносно цього порядку. Порядок на R можна перенести і на множину ДРОБ, якщо порівнювати пари (L, x) за першими компонентами. Це дає змогу говорити про мінімальні елементи в ДРОБ. В алгоритмі МА порядок вибору пар із ДРОБ нефіксований, а тут, скориставшись відношенням (6.3.2), замінимо оператор “Взяти (K, x) із ДРОБ;” оператором “Взяти мінімальний елемент (K, x) із ДРОБ;”. Використовуючи те ж саме співвідношення, оператор “Для всіх L із R виконати;” замінимо оператором “Для всіх L із R , таких, що $L > K$, виконати;”. Оскільки в циклі з елементами ДРОБ щоразу вибирається мінімальний елемент, то для всякого L із R , такого, що $L > K$, має місце властивість, що жодна пара (L, x) не може бути взята раніше пари (K, x) як пара, по якій

ведеться подрібнення. Це дозволяє спростити умовний оператор
“Якщо $(L, x) \in \text{ДРОБ}$, то замінити (L, x) в ДРОБ на $\{(L', x), (L'', x)\}$;”.

В результаті одержуємо алгоритм, орієнтований на обробку ациклических автоматів.

Алгоритм MA-1 (A)

початок.

$R := R_{\text{rang}}$;

$\text{ДРОБ} := R \times X$;

Поки $\text{ДРОБ} \neq \emptyset$ виконувати

 Взяти мінімальний елемент (K, x) із ДРОБ;

 Для всіх L із R таких що $L > K$ виконати

$L' := \{a \in L \mid f(a, x) \in K\}$;

$L'' := \{a \in L \mid f(a, x) \notin K\}$;

 Якщо $L' \neq \emptyset$ і $L'' \neq \emptyset$ то

 Замінити L в R на $\{L', L''\}$;

 Для всіх x із X виконати

 Замінити (L, x) в ДРОБ на $\{(L', x), (L'', x)\}$;

 кц;

 кя;

 кц;

кінець.

Аналізуючи далі отриманий алгоритм, можна побачити, що множини ДРОБ і R поповнюються новими елементами однаковим способом — з точністю до X -співмножника декартового добутку в множині ДРОБ. Це дає змогу уточнити стратегію вибору елементів із ДРОБ шляхом заміни циклу:

Поки $\text{ДРОБ} \neq \emptyset$ виконувати

 Взяти мінімальний елемент (K, x) із ДРОБ;

P ;

 кц;;

де P — оператор, який відповідає тілу циклу, подвійним циклом

Поки $\text{ДРОБ} \neq \emptyset$ виконувати

 Взяти $\{K\} \times X$ із ДРОБ, де K — мінімальний клас у множині ДРОБ;

 Для всіх x' із X виконати P ;

 кц;;

кц;;

Звернемо увагу на таку властивість оператора P : якщо деякий клас L із R при виконанні оператора P вилучається з R , то одночасно з ним із множини ДРОБ вилучаються всі пари виду (L, x) для $x \in X$. Звідси і з того, що початкове значення ДРОБ = $R \times X$, випливає, що в точці, яка йде безпосередньо за оператором “Взяти (K, x) із ДРОБ,” справедливе інваріантне співвідношення “ $K \in R$ ”. Отже, в цій точці можна вставити надлишковий оператор “Відмітити K в R ”. В результаті приходимо до такого алгоритму.

Алгоритм MA-2 (A)

початок.

$R := R_{\text{rang}}$;

ДРОБ := $R \times X$;

Поки ДРОБ $\neq 0$ виконувати

Взяти $\{K\} \times X$ із ДРОБ, де K — мінімальний клас у множині ДРОБ;

Відмітити K в R ;

Для всіх x' із X виконати P ;

кц;

кц;

кінець.

Нехай α — точка входу в тіло зовнішнього циклу, і нехай R' — множина невідмічених елементів із множини R . Тоді легко перевірити, що в точці α існує інваріант “ДРОБ = $R' \times X$ ”. Використовуючи його, умову самого зовнішнього циклу ДРОБ $\neq \emptyset$ замінimo умовою “ $R' \neq \emptyset$ ”, а оператор “Взяти $\{K\} \times X$ із ДРОБ ...;” — оператором “Знайти в R мінімальний невідмічений клас K ;”. Після цих перетворень структура даних ДРОБ та всі пов’язані з її обчисленням оператори стають надлишковими, і їх можна вилучити. В результаті приходимо до такого алгоритму.

Алгоритм MA-3 (A)

початок.

$R := R_{\text{rang}}$;

Поки в R є невідмічені класи виконувати

Знайти мінімальний невідмічений елемент K ;

Відмітити K в R ;

Для всіх x із X виконати

Для всіх L із R таких, що $L > K$, виконати

$L' := \{a \in L \mid f(a, x) \in K\}$;

$L'' := \{a \in L \mid f(a, x) \notin K\}$;

якщо $L' \neq \emptyset$ і $L'' \neq \emptyset$, то Замінити L в R на $\{L', L''\}$;

кя;
 кц;
 кц;
кінець.

І на закінчення, замінюючи зовнішній цикл, оператори “Знайти в $R...$;” і “Відмітити K в $R;$ ” циклом “Для всіх K із R в порядку зростання виконати;”, одержуємо остаточну версію алгоритму мінімізації ациклічних автоматів.

*Алгоритм MAA (A)
 початок.*

$R := R_{\text{rang}};$

Для всіх K із R в порядку зростання виконати

Для всіх x із X виконати

Для всіх L із R таких, що $L > K$ виконати

$L' := \{a \in L \mid f(a, x) \in K\};$

$L'' := \{a \in L \mid f(a, x) \notin K\};$

якщо $L' \neq \emptyset$ і $L'' \neq \emptyset$, то Замінити L в R на
 $\{L', L''\};$

кя;

кц;

кц;

кц;

кінець.

Якісну характеристику побудованого алгоритму МАА дає таке твердження.

Теорема 6.3.2. *Приведення ациклічного автомата A алгоритмом MAA виконується за час $O(m)$, де m — число переходів в автоматі A [20].*

Д о в е д е н н я. Побудова початкового розбиття R_{rang} , пов’язаного з розбиттям станів за рівнями, виконується за один обхід графа переходів автомата A . Розбиття R_{rang} представимо у вигляді списку рівнів, кожний з яких, в свою чергу, теж подається списком класів. Обхід класів виконується від нульового рівня до максимального, а всередині рівня — в порядку переліку класів. При подрібненні класів відносно елементарного подрібнювача (K, x) скористаємося відношенням f^{-1} , оберненим до функції переходів f . За допомогою цього відношення за обмежений час виконується доступ до кожного елемента a , такого, що $f(a, x) \in K$. Знайдений елемент a вилучається із відповідного класу L і заноситься в новий клас L' , який зв’язаний з L . Операції вилу-

чення a із L і додавання його в L' , як відомо, можна виконати за обмежений час, якщо представити класи двосторонніми списками. Після подрібнення всіх класів відносно (K, x) для кожного класу L , який мав елементи, що під дією x переводились в K , виконується перевірка: буде чи ні цей клас пустим після переміщення таких елементів. Якщо він пустий, то L' розглядається як L , а якщо ні, то відповідний рівень поповнюється новим класом — L' . Очевидно, що така перевірка і поповнення списку класів всередині рівня ще одним класом виконується за обмежений час. Тепер для одержання остаточної оцінки досить зауважити, що кожний перехід $f(a, x) = a'$ в алгоритмі МАА використовується при подрібненні класів рівно один раз.

Теорема доведена.

Розглянемо модифікацію функції подрібнення. Ця модифікація визначається для довільних класів L, K і довільних елементів x, x' із X так:

$$L/(K, \{x, x'\}) = \begin{cases} L_1' = \{a \in L \mid f(a, x) \& f(a, x') \in K\}, \\ L_2' = \{a \in L \mid f(a, x) \in K \& f(a, x') \notin K \vee \\ \quad \vee f(a, x) \notin K \& f(a, x') \in K\}, \\ L_3' = \{a \in L \mid f(a, x) \& f(a, x') \notin K\}, \end{cases}$$

де $\&$ і \vee — операції булевої алгебри.

Оскільки операції $\&$ і \vee комутативні, то $L/(K, \{x, x'\}) = L/(K, \{x', x\})$. Іншими словами, два стани із класу L попадають в один і той же клас розбиття в результаті застосування модифікованої функції подрібнення, якщо x і x' — нащадки цих станів — обидва або один із них належить, або жоден із них не належить класу K , незалежно від порядку, в якому беруться елементи x і x' .

Як бачимо, дана модифікація очевидним чином узагальнюється на довільне скінченнє число елементів із вхідного алфавіту X , зокрема

$$L/(K, X) = L/(K, \{x_1, x_2, \dots, x_k\}).$$

Помінявши в алгоритмі МАА внутрішні цикли місцями, приходимо до алгоритму, який обчислює розбиття множини станів автомата A на класи еквівалентності, що відповідає відношенню

$$(a, a') \in L \Leftrightarrow s(a, K) = s(a', K),$$

де L, K — деякі класи розбиття, а $s(a, K)$ — число переходів із стану a в стани з класу K під дією символів із вхідного алфавіту X .

Опишемо реалізацію цієї функції і з'ясуємо її часові характеристики складності. При побудові розбиття за допомогою модифікованої функції подрібнення класи, які виникають в процесі її обчислення, пов'язуються між собою списком в такий спосіб, як це показано на рис. 6.3.4 (тут для $a \in L_i$ маємо $s(a, K) = k - i$, де $i = 1, 2, \dots, k$).



Рис. 6.3.4

При цьому елемент з класу L'_i щоразу переноситься в сусідній клас L'_{i-1} . Адресу сусіднього класу зручно зафіксувати в деякому місці списку L'_i , наприклад у першому елементі. Після завершення обчислення модифікованої функції подрібнення виконується обхід всіх перших елементів списків L'_i і адреси, записані в них, стають нульовими. Крім того, в процесі цього обходу видачаються пусті класи.

Очевидно, що така реалізація модифікованої функції подрібнення вимагає часу $O(k)$, де $k = |X|$.

§ 6.4. АЛГОРИТМИ ПОБУДОВИ КОНГРУЕНТНИХ ЗАМИКАНЬ ДЛЯ СКІНЧЕНИХ АВТОМАТІВ

1. ОЗНАЧЕННЯ ВІДНОШЕННЯ КОНГРУЕНТНОГО ЗАМИКАННЯ

Нехай R — деяке відношення еквівалентності на A . Відношення R^* називається **конгруентним замиканням** відношення R тоді і тільки тоді, коли

$$aR^*a' \Leftrightarrow aRa' \vee (\forall x \in X')(f(a, x) R^* f(a', x)).$$

Очевидно, що відношення R^* є відношенням еквівалентності, але, взагалі кажучи, не є відношенням конгруентності. Дійсно, не завжди

$$aR^*a' \Rightarrow (\forall x \in X)(f(a, x) R^* f(a', x)).$$

Приклад 6.4.1

Нехай задано стани a та a' , такі, що aRa' , і стани $a_1 = f(a, x)$, $a'_1 = f(a', x)$, $a_2 = f(a, x')$, $a'_2 = f(a', x')$, такі, що $a_2Ra'_2$ і не $a_1Ra'_1$. Тоді з aR^*a' не випливає $f(a, x')R^*f(a', x')$ і тому R^* не є відношенням конгруентності (рис. 6.4.1).



Рис. 6.4.1.

Поставимо у відповідність кожному стану a із A вектор цілих чисел $v(a) = (i_1, i_2, \dots, i_k)$, де i_j — номер класу еквівалентності стану $f(a, x_j)$, $j = 1, k$, $k < |X|$. Тоді очевидно, що $v(a) = v(a') \Leftrightarrow aR^*a'$. Цей факт дає можливість дати означення ще одному важливому відношенню: RS^* — симетричне конгруентне замикання відношення R . Дійсно, нехай S — деяка перестановка елементів вектора $v(a) = (i_1, i_2, \dots, i_k)$, тобто $S(v(a)) = v(S(i_1, i_2, \dots, i_k)) = (i'_1, i'_2, \dots, i'_k)$. Тоді

$$aRS^*a' \Leftrightarrow (aRa') \vee (\exists S)(v(a') = v(S(i_1, i_2, \dots, i_k)) = S(v(a))).$$

Інакше кажучи, aRS^*a' тоді і тільки тоді, коли a і a' або ап'ярно еквівалентні, або їх вектори рівні з точністю до порядку переліку компонентів. Для бінарних автоматів зручно вважати, що $X = \{0, 1\}$. Розглядаючи бінарні скінченні автомати, ми не обмежуємо загальності, оскільки всякий автомат із входним алфавітом, який включає більше двох символів, можна перетворити в бінарний автомат. Це досягається шляхом розширення множини станів початкового автомата і кодуванням входних символів в алфавіті $\{0, 1\}$. Таке перетворення завжди можливе, оскільки всяка напівгрупа зі скінченим числом твірних ізоморфно вкладається в напівгрупу, яка породжується двома твірними $\{0, 1\}$.

Приклад 6.4.2

Нехай $X = \{x, x', x''\}$, $A = \{1, 2, 3, 4, 5\}$ (рис. 6.4.2). Закодуємо слова X в алфавіті $\{0, 1\}$. Для цього необхідно закодувати тільки твірні: $x = p = 01$, $x' = p' = 10$, $x'' = p'' = 11$. Тоді одержаний автомат матиме вигляд, наведений на рис. 6.4.3.

Із наведеного прикладу очевидним чином випливає справедливість такого простого твердження.

Твердження 6.4.1. Якщо $A = (A, X, f, F)$ — скінчений автомат і $A' = (A', \{0, 1\}, f', F')$ — бінарний автомат, що йому відповідає, то $|A'| \leq |A| + |A| \cdot (\log |X| + 1) = |A| \cdot (\log |X| + 2)$.

Д о в е д е н н я. Для того щоб закодувати $k = |X|$ елементів, необхідно k слів довжини $\log k + 1$ в алфавіті $\{0, 1\}$. Значить, кожний перехід в автоматі A моделюється не більше, ніж $\log k +$

+ 1 переходами в автоматі A' і для цього необхідно ввести не більше $\log k + 1$ станів. Звідси і випливає наведена оцінка для числа станів.

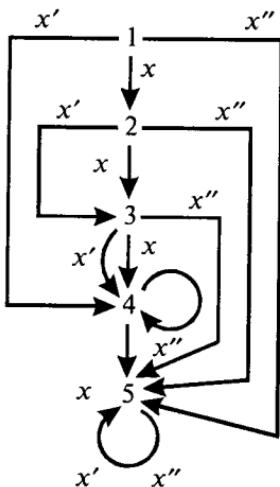


Рис. 6.4.2.
Початковий автомат A

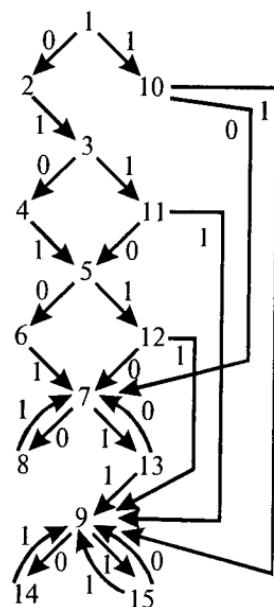


Рис. 6.4.3.
Бінарний автомат A' для A .

2. ЗАГАЛЬНИЙ АЛГОРИТМ ОБЧИСЛЕННЯ КОНГРУЕНТНОГО І СИМЕТРИЧНОГО КОНГРУЕНТНОГО ЗАМИКАНЬ

Перед тим як розглянути алгоритм обчислення конгруентного замикання скінчених X -автоматів, опишемо його структури даних.

Структура T , яку будемо називати *деревом переходів на класах*, являє собою скінчений ациклічний автомат і дозволяє за час $O(|X|)$ визначити стан a' , такий, що aR^*a' .

Для визначення стану a' на вході автомата T подається вектор $v(a) = (i_1, i_2, \dots, i_k)$, і якщо $f_A(1, v(a)) = a'$, то aR^*a' , в протилежному випадку стан a' невизначений, тобто результатом буде 0 ($T(a) = a'$, якщо a' визначений, і $T(a) = 0$, якщо a' невизначений).

Структура pending являє собою змінну типу *множина* і слугує для роботи з множиною станів початкового автомата, а

змінна `combine` — для роботи з множиною пар станів початкового автомата.

Інші структури даних мають таку семантику:

`find(a)`: дає ім'я класу, якому належить стан a ;

`list(e)`: дає список станів, серед яких є хоча б один перехід в стани із класу e ;

`union(e, e')`: функція об'єднання двох класів e і e' в один клас з іменем e .

Узагальнений алгоритм обчислення конгруентного замикання.

УАКЗ (B, R);

початок.

`pending := A; /* A — множина станів автомата A^* */`

поки `pending ≠ ∅` виконувати

`combine := ∅;`

 для кожного a із `pending` виконати

 якщо $T(a) = 0$ то скоректувати переходи в T

 інакше `add (a, T(a))` в `combine` кя

 кц;

`pending := ∅;`

 для кожного (a, a') із `combine` виконати

 якщо `find(a) ≠ find(a')` то

 якщо $|list(find(a))| < |list(find(a'))|$ то

 для кожного b із `list(find(a))` виконати

`add b` в `pending`

 кц;

`union(find(a'), find(a))`

 інакше

 для кожного b із `list(find(a'))` виконати

`add b` в `pending`

 кц;

`union(find(a), find(a'))`

 кя;

 кц;

кц;

кінець.

Якісну характеристику алгоритму УАКЗ дає така теорема.

Теорема 6.4.1. Алгоритм УАКЗ обчислення конгруентного замикання відношення еквівалентності R для автомата $A = (A, X, f, F)$ вимагає часу $O(k \cdot m \cdot \log m)$, де $k = |X|$, $n = |A|$, $m = k \cdot n$.

Д о в е д е н н я. Нехай автомат A заданий таблицею переходів, класи відношення R — одновимірним масивом M довжини

n так, що кожний елемент $M(a)$ має два поля: ім'я поточного класу, якому належить стан a , і ім'я класу, якому належав елемент a раніше (i^h та i^c відповідно).

Автомат T задамо двовимірним масивом $m \times n$ і покажчиком p вільного елемента в цьому масиві (T спочатку нульовий, тобто $p = 1$). Початковим станом автомата T буде стан 1.

Корекція переходів в автоматі T , яка відповідає вектору $v(a) = (i_1^h, i_2^h, \dots, i_k^h)$, виконується за такими правилами.

1. Якщо $T(1, i_1^h) = 0$ і $T(1, i_1^c) = 0$, то покладемо $T(1, i_1^h) = p + 1$ і збільшимо p на одиницю.

2. Якщо $T(1, i_1^h) = 0$ і $T(1, i_1^c) \neq 0$, то покладемо $T(1, i_1^h) = T(1, i_1^c)$, а $T(1, i_1^c) = 0$.

Якщо $T(1, i_1^h) = j \neq 0$, то знаходимо $T(j, i_2^h)$: якщо $T(j, i_2^h) = 0$, виконуємо крок 1 для $T(j, i_2^h)$, а якщо $T(j, i_2^h)$ визначений — крок 2) і т.д.

3. Якщо всі переходи в T скоректовані, а $T(j', i_k^h) = 0$, то $T(j', i_k^h)$ кладемо рівним p , збільшуємо p на 1 і станові $T(j', i_k^h)$ ставимо у відповідність стан a .

Якщо ж стан $T(j', i_k^h)$ визначений і йому відповідає стан a' , то пару (a, a') заносимо в множину *combine*.

З опису кроків 1—3 випливає, що для заданого вектора $v(a)$ корекція переходів вимагає часу, пропорціонального довжині вектора $v(a)$, тобто $O(k)$ одиниць часу.

Тепер для доведення теореми підрахуємо кількість операцій, які виконуються в процесі роботи алгоритму на структурах даних.

Для структури *pending* число операцій *add* при реалізації стратегії *модифікації меншої половини* не перевищує величини $n + k \cdot n \cdot \log k \cdot n = O(m \cdot \log m)$. Оскільки при обробці одного елемента із множини *pending* необхідно затратити $O(k)$ одиниць часу для корекції переходів в автоматі T , то загальна складність роботи з елементами множини *pending* виражається величиною $O(k \cdot m \cdot \log m)$.

Число операцій *add* для множини *combine* обмежене числом операцій над елементами множини *pending*, тобто обмежене величиною $O(m \cdot \log m)$, а значить, і число операцій *find* теж обмежене цією величиною.

При об'єднанні класів використовується алгоритм швидкого об'єднання класів, які не перетинаються. При цьому з кожним класом еквівалентності пов'язуються двосторонні списки станів,

що є безпосередніми попередниками станів даного класу, і число, яке характеризує кількість елементів у цих списках попередників. Оскільки число операцій union обмежене числом n , то загальний час для цих операцій виражається величиною $O(n \cdot G(n))$, де $G(n)$ — функція, обернена до функції Аккермана [15].

Підводячи підсумок, маємо, що загальна складність алгоритму УАКЗ виражається величиною $O(k \cdot m \cdot \log m)$.

Теорема доведена.

Розглянемо роботу алгоритму УАКЗ на прикладі.

Приклад 6.4.3

Нехай дано автомат A , зображений на рис. 6.4.2. Необхідно обчислити конгруентне замикання відношення тотожності i .

Перенумеруємо класи автомата A : $\{1\} = 1$, $\{2\} = 2$, $\{3\} = 3$, $\{4\} = 4$, $\{5\} = 5$. Значить, $M(1).l = 1$, $M(1).p = 0$, $M(2).l = 2$, $M(2).p = 0$, $M(3).l = 3$, $M(3).p = 0$, $M(4).l = 4$, $M(4).p = 0$, $M(5).l = 5$, $M(5).p = 0$.

Множина pending = $\{1, 2, 3, 4, 5\}$, combine = \emptyset . Виберемо довільний елемент з множини pending, наприклад 2. Будуємо вектор $v(2) = (3^h, 3^h, 5^h)$ ($v(2) = (0^c, 0^c, 0^c)$). Виконуємо корекцію переходів в автоматі T для $v(2)$ (рис. 6.4.4 — ліва гілка в дереві T). Далі цей процес повторюємо для станів 1, 3, 4, 5.

В результаті одержали автомат T , зображений на рис. 6.4.4. Безпосередньо з рисунка видно, що стани 3 і 4 еквівалентні, оскільки $T(4) = 3$, тобто $combine = \{(3, 4)\}$.

Після цього pending стає пустою множиною і починає працювати частина алгоритму, яка обробляє елементи із множини combine.

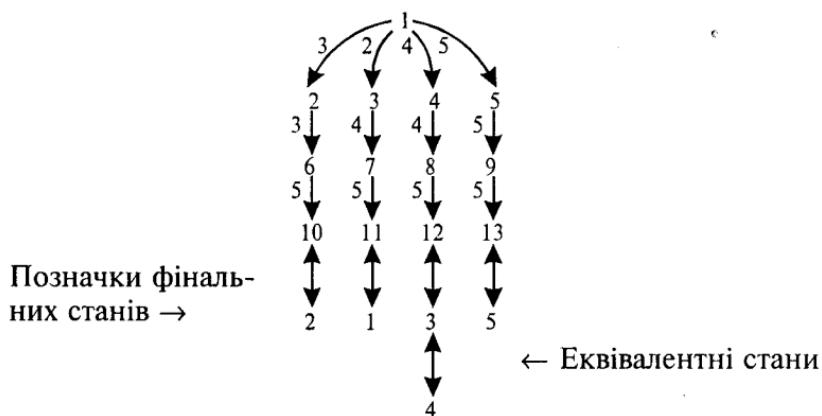


Рис. 6.4.4.

Ацикличний автомат T для векторів класів станів

Беремо пару $(3, 4)$ із combine і знаходимо множину станів $\text{list}(\text{find}(3))$, тобто ті стани, які зв'язані зі станом 3 хоча б одним переходом. Ця множина складається з одного стану — 2. Об'єднуємо стани 3 і 4 в один клас з іменем 4, включаємо стан 2 в множину pending і, оскільки множина combine пуста, починаємо обробляти елементи множини $\text{pending} = \{2\}$.

Коректуємо переходи в T для $v(2) = (4^h, 4^h, 5^h)$ ($v(2) = (3^c, 3^c, 5^c)$). Знаходимо $T(2) = 4$ і пару $(2, 4)$ включаємо в множину combine . Після цього, повторюючи весь процес для пари $(2, 4)$, знаходимо стан 1, для якого $v(1) = (4^h, 4^h, 5^h)$ і $T(1) = 4$. Виконуючи аналогічні дії для стану 1, остаточно одержуємо такі класи еквівалентності: $\{1, 2, 3, 4\} - 4$ і $\{5\} - 5$. \blacktriangleleft

Алгоритм УАКЗ легко узагальнюється на випадок симетричних конгруентних замикань. Дійсно, для обчислення симетричного конгруентного замикання необхідно упорядкувати компоненти вектора $v(a)$ для кожного стану a з A перед тим як визначити клас його еквівалентності. Первінне сортування всіх станів може бути виконане за $O(k \cdot k \cdot n) = O(k \cdot m)$ одиниць часу, а наступні сортування для кожного стану — за $O(k)$ одиниць часу. Для виконання такого сортування використовують процедури sort і ordering .

Алгоритм УАКСЗ для обчислення симетричного конгруентного замикання має такий вигляд.

УАКСЗ (B, R)

початок.

```

pending := A; /* A — множина станів автомата A*/
для кожного a із pending виконати
    sort(v(a));
кц;
поки pending ≠ ∅ виконувати
    combine := ∅;
    для кожного a із pending виконати
        якщо  $T(a) = 0$  то скоректувати переходи в  $T$ 
        інакше add (a,  $T(a)$ ) в combine кя
    кц;
    pending := ∅;
    для кожного (a, a') із combine виконати
        якщо  $\text{find}(a) \neq \text{find}(a')$  то
            якщо  $|\text{list}(\text{find}(a))| < |\text{list}(\text{find}(a'))|$  то
                для кожного b із  $\text{list}(\text{find}(a))$  виконати
                    add b в pending;
                    ordering(v(b), $\text{find}(a)$ , $\text{find}(a')$ );

```

кц;
 union(find(a'), find(a))
 інакше
 для кожного b із list(find(a')) виконати
 add b в pending;
 ordering($v(b)$, find(a'), find(a));
 кц;
 union(find(a), find(a'))
 кя;
 кц;
 кц;
кінець.

Зі сказаного випливає справедливість такого твердження.

Теорема 6.4.2. Часова складність алгоритму обчислення симетричного конгруентного замикання RS^* для скінченного автомата $A = (A, X, f, F)$ відносно апріорного відношення еквівалентності R пропорціональна величині $O(k \cdot m \cdot \log n)$.

Коли ж не всі стани з A задовольняють симетричне відношення, а лише деякі, наприклад стани з деякої підмножини A' , то даний алгоритм теж можна застосувати. Дійсно, при цьому сортуються не всі стани з A , а лише стани з A' . Якщо A' задати у вигляді характеристичного вектора, то оцінка теореми 6.4.2 залишається без змін.

Приклад 6.4.4

Повернемося до автомата, який розглядався вище в прикладі 6.4.3. Нехай переходи цього автомата змінені таким чином:

$$\begin{aligned} v(1) &= (5, 2, 4), \quad v(4) = (4, 5, 4), \\ v(2) &= (5, 3, 3), \quad v(5) = (5, 5, 5), \\ v(3) &= (4, 4, 5), \end{aligned}$$

1. Нехай $A' = A$. Тоді упорядковуємо всі $v(a)$:

$$\begin{aligned} v(1) &= (2, 4, 5), \quad v(4) = (4, 4, 5), \\ v(2) &= (3, 3, 5), \quad v(5) = (5, 5, 5). \\ v(3) &= (4, 4, 5), \end{aligned}$$

Дерево переходів на класах матиме вигляд, поданий на рис. 6.4.4. В результаті роботи УАКСЗ одержуємо, що $RS^* = \{(1, 2, 3, 4), (5)\}$.

2. Нехай $A' = (1, 3, 4)$. Тоді дерево переходів T автомата B має вигляд, поданий на рис. 6.4.5. В результаті роботи УАКСЗ знаходимо, що $RS^* = \{(1), (2), (3, 4), (5)\}$. \blacktriangleleft

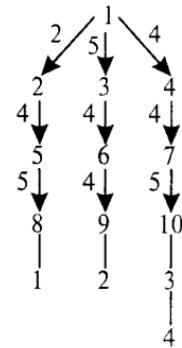


Рис 6.4.5.
Дерево переходів

Алгоритм УАКСЗ легко трансформується в алгоритм конгруентного замикання для бінарних автоматів [83]. Для них робота зі структурою T може бути організована більш ефективно. Дійсно, кожний вектор деякого стану a автомата A являє собою пару $v(a) = (i, j)$, і ми можемо покласти $T(a) = T(i, j) = a'$, якщо a' визначено, і 0, якщо a' не визначено. Отже, у цьому випадку час обчислення стану a' є константою, і це дає змогу покращити оцінку часової складності алгоритму як УАКЗ, так і УАКСЗ.

Наслідок 6.4.1. Часова складність алгоритмів УАКЗ і УАКСЗ для бінарного автомата A виражається величиною $O(m \cdot \log m)$, де m — число переходів в автоматі A .

З твердження 6.4.1 випливає такий наслідок.

Наслідок 6.4.2. Часова складність алгоритму обчислення конгруентного замикання відношення R для автомата A пропорціональна величині $(m \cdot \log k) \cdot \log (m \cdot \log k)$.

Якщо автомат частковий, то задачу обчислення R^* або RS^* можна звести до попередньої, якщо ввести для всіх a і x , таких, що $f(a, x) = w$, де w — стан невизначеності, спеціальний стан \bar{a} , яким розширяється множина A і для якого $(\forall x \in X) f(a, x) = \bar{a}$.

З цього зауваження випливає, що даний алгоритм можна застосувати і до ациклических автоматів, які є частковими. Але таке довизначення функції переходів може значно збільшити число переходів у початковому автоматі, що, в свою чергу, може негативно вплинути на ефективність алгоритму в цілому.

Розглянемо інший можливий шлях роботи алгоритму УАКЗ з частковими автоматаами. Спочатку скоректуємо означення R^* і RS^* на випадок часткових автоматів. Таку корекцію необхідно зробити, оскільки безпосередньо скористатися векторами $v(a)$ вже неможливо. Означення коректується таким чином:

$$\begin{aligned} aR^*a' &\Leftrightarrow aRa' \vee ((\forall x \in X) (f(a, x) = \\ &= w \& f(a', x) = w) \vee (f(a, x)R^*f(a', x))), \end{aligned}$$

тобто стани $f(a, x)$ і $f(a', x)$ або обидва невизначені, або визначені і лежать в одному класі еквівалентності відношення R^* . Поставимо у відповідність стану a вектор $v(a) = (x_{i_1}, x_{i_2}, \dots, x_{i_l}, j_1, j_2, \dots, j_l)$, де $x \in X$, такий, що $f(a, x)$ лежить у класі з іменем j_p , $p = 1, 2, \dots, l$. Тепер алгоритм УАКЗ можна застосувати до автомата A з векторами $v(a)$. Тоді, як і раніше, маємо $aR^*a' \Leftrightarrow aRa' \vee v(a) = v(a')$.

Якщо обчислюється RS^* , то $aRS^*a' \Leftrightarrow aRa' \vee (\exists S) v'(a) = S(v'(a')) = (x_{i_1}, x_{i_2}, \dots, x_{i_l}, S(j_1), S(j_2), \dots, S(j_l))$, де S — перестановка на іменах класів. Таким чином, для обчислення RS^* , як і раніше, необхідно упорядкувати другу половину вектора $v'(a)$.

Побудова векторів $v(a)$ виконується за час $O(m)$, де m — число переходів в автоматі A , якщо A представляти графом переходів. Коли ж A задавати таблицею переходів, то цей час виражається величиною $O(k \cdot n)$, де $k = |X|$, а n — число станів автомата A .

Якщо вектори обчислюються за час $O(m)$, то алгоритм УАКЗ не змінюється, за винятком автомата T , який у цьому випадку займає вдвічі більше пам'яті, ніж у попередньому. Звідси випливає, що часові характеристики алгоритму не змінюються.

3. АЛГОРИТМ ОБЧИСЛЕННЯ КОНГРУЕНТНИХ ЗАМИКАНЬ ДЛЯ АЦІКЛІЧНИХ АВТОМАТІВ

Якщо X -автомат $A = (A, X, f)$ ациклічний, то можна побудувати більш ефективний алгоритм обчислення конгруентного замикання для такого автомата. Введемо деякі необхідні поняття.

Нехай R — відношення еквівалентності на множині станів автомата A . Говорять, що автомат A над R має цикл, якщо існує така послідовність станів $a = b_1, b_2, \dots, b_{2k+1} = a$, що $f(b_{2i-1}, x) = b_i$ для деякого x із X і $(a_i, a_{i+1}) \in R$ (рис. 6.4.6).

Автомат A над R називається **ациклічним**, якщо A над R не має циклів. Розглянемо випадок, коли A над R і A над R^* ациклічні. Якщо автомат ациклічний, то множину класів відношення R можна упорядкувати за рівнями так, що стани із класів верхнього рівня недосяжні зі станів класів нижнього рівня. Цей порядок може порушитися в результаті злиття класів різних рівнів.

Нехай K — деякий клас розбиття R . Клас K будемо називати фікованим, якщо класи еквівалентності будь-якого нащадка всякого стану a із K не можуть змінитися в процесі обчислень (ці класи фіковані), тобто вектор $v(a)$ довільного елемента із K незмінний в процесі обчислення R^* . Фікованими класами будуть, наприклад, класи розбиття R , які мають найнижчий рівень. Дійсно, стани з цих класів не мають нащадків, і тому їх класи не можуть змінитися в процесі обчислень. Зауважимо, що коли зливаються два фіковані класи, то в результаті знову одержуємо фікований клас, і якщо хоча б один з класів не фікований, то результат теж буде не фікованим класом.

Для обчислення класів розбиття R^* скористаємося функцією подрібнення класів (див. § 6.3). При побудові розбиття множини

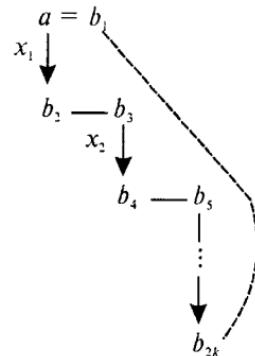


Рис 6.4.6

станів на класи за допомогою функції подрібнення елементарним дільником береться пара (K, x) , де $x \in X$, а K — деякий фіксований клас. Виконавши подрібнення для всіх таких пар (K, x) , можна починати пошук і побудову інших фіксованих класів. Внаслідок ідемнності функції подрібнення класи K з подальшого розгляду можна вилучити.

Таким чином, стратегія обчислень така: доки є фіксовані класи, виконувати подрібнення; використані в процесі обчислення фіксовані класи вилучити з подальшого розгляду; побудувати нові фіксовані класи, і, якщо вони є, повторити процес для нових фіксованих класів. Отже, при такій стратегії необхідно ефективно виконувати операції злиття класів і пошуку фіксованих класів.

Для виявлення фіксованих класів необхідно проглянути всі класи, які зливаються, для визначення того, що жоден із них не зливається з нефіксованим класом. Якщо виявляється, що який-небудь клас не фіксований, то всі фіксовані класи, які з цим класом повинні зливатися, зливаються, і одержаний клас відкладається до тих пір, поки не стане фіксованим. Для зберігання відкладених класів служить множина НЕГОТОВІ в наведеному нижче алгоритмі АКЗАА.

Якщо ж всі класи, що зливаються, фіксовані, то формується новий клас і заноситься в множину ДРОБ — множину елементарних подрібнювачів. Для зберігання “заготовок” для виявлення фіксованих класів служить множина ГТОВІ, а для визначення того, що клас фіксований для кожного стану автомата A , заводиться лічильник числа непройдених переходів із цього стану. Ці лічильники змінюють свої значення в процесі подрібнення класів, а початкові значення лічильників знаходяться, очевидно, за один обхід автомата A , тобто за час $O(m)$, де m — число переходів в автоматі A . Зміна значення $r(a)$ лічильника стану a виконується таким чином:

якщо $a \in L \cap L'$ і L подрібнюється по (K, x) , то коли $a \in L' = \{a \in L \mid f(a, x) \in K\}$, величина $r(a)$ зменшується на одиницю;

якщо $a \in L \setminus L' = \{a \in L \mid f(a, x) \notin K\}$, то $r(a)$ не змінюється.

Таким чином, якщо $r(a) = 0$, то класи нашадків стану a уже фіксовані, в протилежному випадку — не фіксовані. Якщо деякий клас K відкладається, то це значить, що існує $a \in K$, такий, що $r(a) \neq 0$. Отже, якщо автомат A над R і над R^* ациклічний, то стан a в процесі подальших обчислень обов'язково повинен попасті в деякий фіксований клас. Коли таке не трапляється, то A над R^* має цикл. Ця обставина може служити умовою перевірки того, що A над R^* ациклічний.

Розглянемо алгоритм обчислення конгруентного замикання ациклических автоматів.

АКЗАА (A, R)

початок.

ДРОБ := $\{K \in R \mid \forall a \in K r(a) = 0\}$;

РОЗБ := $A \setminus \text{ДРОБ}$;

НЕГОТОВІ := РОЗБ;

Поки ДРОБ $\neq \emptyset$ виконувати

 Для всіх K із ДРОБ виконати

 Для всіх x із X виконати

 Для всіх L із РОЗБ виконати

$L' := \{a \in L \mid f(a, x) \in K\}$;

$r(L') := r(L') - |L'|$;

$L'' := \{a \in L \mid f(a, x) \in K\}$;

 Якщо $L' \neq \emptyset$ & $L'' \neq \emptyset$ то

 (Замінити L в РОЗБ на $\{L', L''\}$;

 Якщо $r(L') = 0$ то ГОТОВІ :=

 ГОТОВІ $\cup \{L'\}$ кя;

 Якщо $r(L'') = 0$ то ГОТОВІ :=

 ГОТОВІ $\cup \{L''\}$ кя;

 кя;

 кц;

 кц;

ДРОБ := \emptyset ;

Поки ГОТОВІ $\neq \emptyset$ виконувати

 Взяти K із ГОТОВІ;

 Формуй \tilde{K} по K ;

 Якщо $r(K) = 0$ то (ДРОБ := ДРОБ $\cup \{\tilde{K}\}$;

 НЕГОТОВІ := НЕГОТОВІ $\setminus \{\tilde{K}\}$)

 інакше НЕГОТОВІ := НЕГОТОВІ $\cup \{\tilde{K}\}$ кя;

 кц;

 кц;

 Якщо НЕГОТОВІ $\neq \emptyset$ то стоп /* цикл */;

кінець.

У наведеному алгоритмі використовується оператор “Формуй \tilde{K} по K ”, який перевіряє фіксованість всіх класів із множини ГОТОВІ. Якщо виявляється, що деякий клас чи сукупність класів із множини ГОТОВІ зв’язані з деяким нефіксованим класом

K' , то даний клас чи сукупність класів вилучається з множини ГОТОВІ з відповідною відміткою станів у цих класах. Відмітка служить для того, щоб по необхідності повторно відкласти клас чи сукупність класів, дані стани уже не будуть аналізуватися, аналізуватимуться лише ті стани, які ще ні разу не аналізувалися. Таким чином, використання такої відмітки дозволяє запобігти повторному аналізу станів у класах.

Якщо ж на деякому кроці виявляється, що клас K фіксований, і всі класи, які з ним пов'язані, теж фіксовані, то за цим класом формується новий клас, який одержує ім'я і заноситься в множину ДРОБ. Має місце таке твердження.

Теорема 6.4.3. Часова складність алгоритму АКЗАА для ациклічного автомата A виражається величиною $O(m)$, де m — число переходів в автоматі A [83, 36].

Д о в е д е н н я пропонується як вправа.

Зауважимо, що коли в алгоритмі АКЗАА використовувати модифіковану функцію подрібнення (див. § 6.3), то за допомогою даного алгоритму можна обчислювати симетричні конгруентні замикання ациклічних автоматів. При цьому часові характеристики алгоритму дещо погіршуються і виражаються величиною $O(k \cdot m)$, де $k = |X|$, а m — число переходів в автоматі, тобто має місце такий наслідок.

Наслідок 6.4.3. Час, необхідний для обчислення алгоритмом АКЗАА симетричного конгруентного замикання деякого відношення еквівалентності для ациклічного автомата A , виражається величиною $O(km)$.

Контрольні питання

1. Дайте означення відношення конгруентного замикання.
2. Дайте означення відношення симетричного конгруентного замикання.
3. Яку часову складність має алгоритм УАКЗ?
4. Яку часову складність має алгоритм УАКСЗ?
5. Які структури даних використовуються в алгоритмі УАКЗ?

Задачі і вправи

1. Дайте доведення теореми 6.3.1.
2. Обґрунтуйте, що ациклічний автомат не може бути повністю визначеним.

3. Побудуйте приклад, який показує, що алгоритм побудови конгрунтного замикання тотожного відношення еквівалентності для заданого автомата і алгоритм мінімізації цього автомата дають різні результати.

4. Побудуйте приведений автомат для автомата A , зображеного на рис. 6.4.2.

5. Доведіть, що алгоритм УАКЗ правильно обчислює результат, тобто, що алгоритм УАКЗ будує розбиття множини станів на класи, які відповідають відношенню еквівалентності R^* .

Р о з д і л 7

МОДЕЛІ АЛГОРИТМІВ І ПРОГРАМ

У даному розділі розглядаються як класичні, так і сучасні математичні моделі алгоритмів і програм. Хоча ці моделі і не зовсім адекватно відображають всі властивості алгоритмів і програм, оскільки будуються шляхом абстрагування, але за їх допомогою можна встановити багато корисних властивостей.

§ 7.1. МАШИНИ ТҮЮРІНГА

1. ОЗНАЧЕННЯ І ФУНКЦІОНУВАННЯ

Розглянемо детальніше поняття *машини Тьюрінга*. Нехай $X = \{x_1, x_2, \dots, x_n\}$ — скінчений алфавіт і $F(X)$ — вільний моноїд всіх слів в алфавіті X . Функцію $f: F(X)^n \rightarrow F(X)$ називають *n-арною словесною функцією*.

Машина Тьюрінга задає *словесну* функцію над алфавітом X і являє собою п'ятірку об'єктів $(X, Q, q_0, \#, P)$ і правило функціонування, спільне для всіх машин, де:

X — *алфавіт* машини;

Q — скінчена непуста множина, елементи якої називаються *станами*;

$q_0 \in Q$ — виділений стан, який називають *початковим*;

$\#$ — спеціальний “пустий” символ, $\# \notin X \cup Q$;

P — *програма* машини.

Програма машини — це скінчена множина слів, що мають вигляд $qxq'x'd$ і називаються *командами*, де $q, q' \in Q$, $x, x' \in X \cup \{\#\}$, а $d = \{l, s, r\}$ — символи напрямку руху для управління поведінкою машини, причому $l, s, r \notin X \cup Q$. Крім того, вважається, що в програмі P жодні дві команди не можуть мати одинакові пари перших двох елементів.

Правило функціонування пояснимо неформально на добре відомій *фізичній моделі* машини Тьюрінга. Машина складається з

потенційно нескінченної (в обидва кінці) стрічки, керування і голівки, яка може переміщуватися вздовж стрічки (див. рис. 3.1.1). Стрічка розбита на клітинки, в яких можуть бути записані символи з алфавіту X , або символ $\#$ — пустий символ. Керування на кожному кроці роботи машини знаходитьться в одному із станів множини Q , розшифровує програму, яка однозначно визначає поведінку машини і керує голівкою. Голівка в кожний момент часу знаходиться проти деякої клітинки стрічки і може сприймати (читати) символи із стрічки, записувати їх на стрічку і переміщуватися вздовж стрічки. Машина функціонує таким чином. В початковий момент часу на стрічці записане деяке початкове слово p із $F(X)$, а керування знаходиться в початковому стані q_0 . Пошукове слово і ті слова, що з'являються в процесі роботи машини, обмежені з обох боків пустими символами $\#$. Голівка знаходиться проти клітинки, де записаний перший символ слова p .

Робота машини полягає в повторенні такого циклу елементарних дій:

- 1) читається символ, записаний у клітинці проти голівки;
- 2) виконується пошук застосованої команди, тобто такої команди $qxq'x'd$, в якій q — поточний стан керування, а x — прочитаний символ;
- 3) виконується команда, що спричиняє переход керування в стан q' та запис в клітинку, що знаходиться проти голівки, символу x' (символ x витирається). Після цього голівка переміщується ліворуч на одну клітинку, якщо $d = l$, або праворуч на одну клітинку, якщо $d = r$, або залишається на місці, якщо $d = s$.

Машина зупиняється лише в тому випадку, коли на черговому кроці роботи жодну з команд не можна застосувати. Слово, що лишилося на стрічці, є результатом роботи машини. Це слово називається **заключним**. Машина Тьюрінга, переробляючи початкові слова в заключні, визначає деяку словесну функцію, для якої початкові слова є аргументами, а заключне слово — значенням цієї функції. Ясно, що робота машини може закінчитися через деякий час, а може і ніколи не закінчитися. Якщо машина Тьюрінга припиняє свою роботу (з деяким словом на стрічці), то функція, що обчислюється цією машиною, вважається визначеною, а якщо машина не зупиняється, то значення відповідної функції вважається невизначенім. При інтерпретації заключного слова як значення функції пусті символи $\#$ ігноруються. Таким чином, машина Тьюрінга T задає деяку часткову функцію f_T і спосіб її обчислення.

Функція f називається **частково обчислюваною за Тьюрінгом**, якщо існує така машина Тьюрінга T , що $f = f_T$. У цьому випадку

також говорять, що для функції f існує (частковий) алгоритм обчислення її значень, який визначається машиною T .

Машина Тьюрінга являє собою формальну модель алгоритму. Дійсно, для неї характерні такі властивості, як дискретність, детермінованість, елементарність кроків, направленість і масовість (див. § 3.1). Зазначимо також, що для машин Тьюрінга характерні:

- конструктивність** — машина являє собою скінчений об'єкт, побудований за цілком визначеними правилами;
- скінченність** — процес обчислення значень функції f_T , для яких вона визначена, складається із скінченного числа кроків;
- однозначність** — результат роботи машини однозначно визначається початковим словом.

Приклад 7.1.1

Нехай $X = \{(), 0, 1, *\}$ і $p \in F(X)$, де $F(X)$ складають ті і тільки ті слова в алфавіті X , що побудовані з перших двох символів X . Правильними словами називаються:

- e — пусте слово;
- якщо α — правильне слово, то слова (α) , $()\alpha$, $\alpha()$ — теж правильні.

Нехай $M \subset F(X)$ означає множину правильних слів. Побудуємо машину T , на стрічку якої записуються слова із $F(X)$. T зупиняється з результатом 1 на стрічці, якщо слово $p \in M$, і з результатом 0 в протилежному випадку. Іншими словами, T повинна обчислювати значення предиката $\pi: F(X) \rightarrow \{0, 1\}$, такого, що

$$\pi(p) = \begin{cases} 1, & \text{якщо } p \in M, \\ 0, & \text{якщо } p \notin M. \end{cases}$$

Шукана машина Тьюрінга має такий вигляд:

$$T = (((), 0, 1, *), \{q_0, q_1, q_2, q_3, q_4\}, q_0, \#, P),$$

де програма P має такі команди:

- | | |
|----------------------|-----------------------|
| 1. $q_0 (q_0 (r,$ | 2. $q_0) q_1 * l,$ |
| 3. $q_0 * q_0 * r,$ | 4. $q_0 \# q_2 \# l,$ |
| 5. $q_0 (q_0 * r,$ | 6. $q_1 * q_1 * l,$ |
| 7. $q_1 \# q_4 0r,$ | 8. $q_2 (q_3 \# l,$ |
| 9. $q_2 * q_2 \# l,$ | 10. $q_2 \# q_2 1s,$ |
| 11. $q_3 (q_3 \# l,$ | 12. $q_3 * q_3 \# l,$ |
| 13. $q_3 \# q_3 0s,$ | 14. $q_4 (q_4 \# r,$ |
| 15. $q_4) q_4 \# r,$ | 16. $q_4 * q_4 \# r.$ |

Розглянемо роботу машини T на словах $p_1 = (())$, $p_2 = (()$. Маємо

$$\begin{aligned}
 T(p_1) &= q_0(()) = (q_0()) = ((q_0)) = (q_1(*)) = (**q_0) = (*q_1** = \\
 &= (q_1*** = q_1(** = *q_0*** = **q_0** = ***q_0* = \\
 &= ****q_0\# = ***q_2* = **q_2* = *q_2* = q_2* = q_2\# = q_21).
 \end{aligned}$$

До слова 1, записаного на стрічці, жодна з команд T не застосовна, отже, слово 1 — результат роботи машини T над словом p_1 . Для слова p_2 маємо

$$\begin{aligned}
 T(p_2) &= q_0(()) = (q_0()) = ((q_0)) = (q_1(* = (*q_0* = (**q_0* = \\
 &= (**q_0\# = (*q_2* = (q_2* = q_2\# = q_30.
 \end{aligned}$$

У даному випадку результатом роботи є слово 0. \blacktriangleleft

Функціонування машини Тьюрінга можна описати за допомогою протоколу роботи над заданим початковим словом. Нехай $#y_1\dots y_k y_{k+1}\dots y_m\#$ — слово, що виникає на стрічці в процесі роботи машини після виконання деякої команди машини, в результаті якої машина знаходиться в стані q , а голівка стоїть проти клітинки стрічки, де записаний k -й символ слова. Слово $#y_1\dots y_k q y_{k+1}\dots y_m\#$ називається **конфігурацією машини T** . Послідовність конфігурацій, записаних у тому порядку, в якому вони з'являються в процесі роботи машини, називається **протоколом роботи машини T** . В наведеному вище прикладі процес роботи машини Тьюрінга описувався за допомогою протоколів.

2. СЛОВЕСНЕ ПРЕДСТАВЛЕННЯ МАШИНИ ТЬЮРІНГА

Машина Тьюрінга однозначно задається своєю програмою. Якщо упорядкувати деяким способом її команди і застосувати описаний нижче спосіб кодування послідовності команд словом в алфавіті машини Тьюрінга, то можна одержати її представлення у власному алфавіті.

Нехай X — алфавіт машини Тьюрінга T , а Q — множина її станів. Упорядкуємо деяким чином множину Q . Нехай $K(q)$ — порядковий номер стану q . Введемо додатковий символ $*$, що не входить в X , і поставимо у відповідність кожній команді $qaq'a'd$ слово в алфавіті $W = X \cup \{\#, l, r, p, *\}$:

$$*^{K(q)} a *^{K(q')} a' d.$$

Упорядкованій послідовності команд відповідає послідовність слів в алфавіті W . Результатом конкатенації слів послідовності є слово p_T , що однозначно описує машину T . Наступний етап кодування — це перехід від представлення машини T в алфавіті W до представлення її в алфавіті X . Якщо алфавіт X має n символів,

то, упорядкувавши його деяким чином, упорядкуємо алфавіт W так, щоб додаткові символи, які не входять в X , одержали такі номери: $K'(\#) = n + 1$, $K'(*) = n + 2$, $K'(l) = n + 3$, $K'(r) = n + 4$, $K'(s) = n + 5$.

Закодуємо слова з $F(X)$ і $F(W)$ числами, використовуючи лексикографічний порядок (див. розділ 9, § 9.1). Знайдемо номер слова p_T із $F(W)$ і, вибравши слово з $F(X)$ з тим же номером, одержимо словесне представлення α_T машини Тьюрінга в її алфавіті. За словом α_T можна однозначно (з точністю до номерів станів) відновити програму машини T . Слід зазначити, що одній і тій же машині Тьюрінга можуть відповідати різні словесні представлення в її алфавіті залежно від способу упорядкування множин Q , X і P , але за будь-яким із таких представлень програма машини відновлюється однозначно.

3. АЛГОРИТМІЧНО РОЗВ'ЯЗНІ І НЕРОЗВ'ЯЗНІ ПРОБЛЕМИ

Нехай X — алфавіт, $M \subseteq F(X)$ — деяка множина слів у цьому алфавіті.

Характеристичною функцією множини M називається повністю визначений на $F(X)$ предикат

$$\chi : M \rightarrow \{0, 1\},$$

що задається таким чином:

$$\chi(p) = \begin{cases} 1, & \text{якщо } p \in M, \\ 0, & \text{якщо } p \notin M. \end{cases}$$

Частковою характеристичною функцією множини M називається функція $\chi : M \rightarrow \{1\}$, яка визначається так:

$$\chi(p) = \begin{cases} 1, & \text{якщо } p \in M, \\ \text{не визначена,} & \text{якщо } p \notin M. \end{cases}$$

Множина M називається **розв'язною**, якщо її характеристична функція обчислена, тобто частково рекурсивна, і **переліченою**, якщо її часткова характеристична функція обчислена. Розв'язність множини M означає, що існує така машина Тьюрінга, яка завжди зупиняється і дає можливість за скінченне число кроків встановити, належить чи ні слово p із $F(X)$ множині M . Для переліченої множини існує така машина Тьюрінга, що зупиняється тоді і тільки тоді, коли слово p належить множині M . У випадку переліченої множини машина Тьюрінга не дає можливості точно встановити, належить чи ні задане слово множині M ,

оскільки відсутність відповіді до деякого моменту часу не несе ніякої інформації про те, з'явиться чи ні вона пізніше. Але коли машина зупинилася, ми точно знаємо, що дане слово належить множині M .

Зв'язок між розв'язними і переліченими множинами виявляє теорема Поста.

Теорема Поста. *Множина $M \subseteq F(X)$ є розв'язною тоді і тільки тоді, коли M і її доповнення $M' = F(X) \setminus M$ переліченні.*

Д о в е д е н н я. Із розв'язності M випливає її переліченність. Дійсно, машину Тьюрінга T_M , що обчислює характеристичну функцію множини M , легко перетворити в машину Тьюрінга $T_{M'}$, додавши до програми машини T команди, які зациклюють її, коли вона зупиняється з результатом 0. Нова машина задає часткову характеристичну функцію множини M .

Нехай T_M і $T_{M'}$ — машини Тьюрінга, що визначають часткові характеристичні функції множин M і M' . Алгоритм, який розпізнає, належить чи ні слово r множині M , зводиться до такої процедури. Виконується по одній команді кожної машини Тьюрінга з одним і тим же словом r на стрічці. Якщо після виконання однієї команди зупиняється машина T_M , то результат роботи алгоритму — символ 1. Якщо зупиняється машина $T_{M'}$, то результат — символ 0. Якщо жодна з машин не зупиняється, то виконується наступна пара команд, і т. д. Оскільки r — це елемент або з M , або з M' , то через скінченне число кроків або T_M , або $T_{M'}$ зупиниться. Таким чином, алгоритм обчислює значення характеристичної функції множини M , і можна побудувати машину Тьюрінга T_M , що реалізує цей алгоритм. Отже, множина M розв'язна.

Теорема доведена.

Означення розв'язної і переліченої множин можна перенести і на числові множини і множини, елементами яких є об'єкти складнішої структури. Прикладами розв'язних можуть бути такі множини:

- пуста множина;
- множина всіх слів у деякому алфавіті X ;
- множина всіх словесних представлень машин Тьюрінга над алфавітом X .

Перейдемо тепер до знайомства із множинами, які не є розв'язними і переліченими.

Проблема зупинки. Машина Тьюрінга, працюючи над деяким словом r із $F(X)$, може як завершити свою роботу, так і не завершити її. Корисно було б мати алгоритм, який для всякого слова r із $F(X)$ з'ясовував би, зупиниться чи ні машина Тьюрінга при роботі зі словом r . Проблему зупинки можна сформулювати

також у термінах множин. Нехай M — множина всіх пар слів в алфавіті X , де в кожній парі перше слово — це словесне представлення деякої машини Тьюрінга, а друге — таке, на якому машина, почавши роботу над ним, зупиняється. Проблема зупинки тепер має такий вигляд: чи є множина M розв'язною?

Теорема Тьюрінга. *Проблема зупинки машини Тьюрінга не є алгоритмічно розв'язною.*

Доведення. Необхідно встановити, чи є обчисленною характеристична функція $g_M: F(X) \rightarrow \{0, 1\}$. Припустимо, що це так і T_g — машина Тьюрінга, що обчислює цю функцію. З обчисленності функції g_M і розв'язності множини M словесних представлень машин Тьюрінга в алфавіті X випливає обчисленність часткової унарної функції $G: F(X) \rightarrow \{0, 1\}$, що задається таким чином:

$$G(p) = \begin{cases} g_M(p, p) & \text{для всіх } p \in M, \\ \text{не визначена} & \text{для всіх } p \notin M. \end{cases}$$

Отже, функція G зв'язує машини Тьюрінга зі своїми власними словесними представленнями.

Введемо ще одну унарну функцію $K: F(X) \rightarrow \{0\}$, де

$$K(p) = \begin{cases} G(p), & \text{якщо } G(p) = 0, \\ \text{не визначена} & \text{в протилежному випадку.} \end{cases}$$

Ясно, що функція K обчисленна, якщо обчисленна функція G . Дійсно, машина Тьюрінга T_K повністю збігається з машиною Тьюрінга T_G , крім того, що, коли машина T_G зупиняється з результатом 1, тоді машина T_K зациклюється.

Нехай β_K — словесне представлення машини T_K в алфавіті X . Спробуємо вияснити, визначено чи ні значення $K(\beta_K)$, тобто спробуємо розв'язати проблему зупинки для пари: машина T_K і її словесне представлення. Припустимо, що значення $K(\beta_K)$ не визначено, тобто машина T_K не зупиняється, якщо вона почала роботу над словом β_K . Тоді $g_M(\beta_K, \beta_K) = 0$, $G(\beta_K) = 0$ і $K(\beta_K) = 0$, тобто значення $K(\beta_K)$ визначено. А це суперечить припущеню. Припустимо тепер, що значення $K(\beta_K)$ визначено, тобто машина T_K зупиняється, почавши роботу над словом β_K . Тоді $g_M(\beta_K, \beta_K) = 1$, $G(\beta_K) = 1$ і $K(\beta_K)$ не визначено, що також суперечить припущеню.

Обидва припущення щодо функції K приводять до суперечності, що спростовує гіпотезу про обчисленність функції g_M і розв'язність множини M .

Теорема доведена.

Доведена теорема встановлює існування функцій, які не є

обчисленими (характеристична функція g_M). З тези Черча випливає, що для такої функції не існує алгоритму обчислення її значень. Але ця теорема не виключає можливості, що проблема зупинки може виявитися розв'язною для деякого вужчого класу машин Тьюрінга.

Проблема пустої стрічки. Розглянемо окремий випадок проблеми зупинки. З'ясуємо, чи є розв'язною проблема зупинки машини Тьюрінга, яка застосовується до пустої стрічки, тобто до стрічки, яка включає лише пустий символ $\#$. Іншими словами, необхідно вияснити, розв'язна чи ні множина M словесних представлень всіх машин Тьюрінга; які зупиняються, почавши роботу при пустій стрічці.

Теорема 7.1.1. *Проблема пустої стрічки не є алгоритмічно розв'язною.*

Доведення. Кожній парі (T, p) , де T — деяка машина Тьюрінга, а p — слово в її алфавіті, покладемо у відповідність машину T_p , програма якої збігається з програмою машини T , крім того, що, почавши роботу, машина T_p витирає початкове слово на стрічці і записує замість нього слово p . Конструкція машини T_p очевидна: до програми машини T необхідно додати відповідним чином команди машини з вправи 6, що наводиться в кінці параграфа.

Машина T_p , почавши роботу з пустою стрічкою, веде себе після запису слова p так, як і машина T , що застосовується до слова p . Зокрема, машина T_p зупиниться в тому і тільки в тому випадку, коли зупиниться машина T . Припустимо, що проблема зупинки машини Тьюрінга з пустою стрічкою розв'язна. Ale тоді звідси випливало б розв'язність проблеми зупинки в загальному випадку. Дійсно, для того щоб дізнатися, зупиняється чи ні деяка машина T , яка застосовується до слова p , необхідно сконструювати машину T_p . З'ясувавши, зупиняється чи ні T_p при пустій стрічці, ми тим самим дізналися б, зупиняється чи ні машина T . A це суперечить теоремі Тьюрінга і спростовує припущення про розв'язність проблеми пустої стрічки.

Теорема доведена.

При доведенні цієї теореми використовувалась така схема: виконавши деякі допоміжні побудови, ми припускаємо розв'язність проблеми, що досліджується, і одержуємо можливість розв'язати іншу проблему, про яку відомо, що вона не є розв'язною. Одержана суперечність доводить нерозв'язність проблеми, що досліджується. Такий спосіб доведення називається **методом зведення**: відома нерозв'язна проблема зводиться до проблеми, що нас цікавить, а остання не є розв'язною. Цей метод застосовується і у випадку, коли треба довести, що проблема, яка нас

цікавить, не є частково розв'язною. Але слід зазначити, що метод зведення не є універсальним: існують нерозв'язні проблеми, які не зводяться одна до другої.

Проблема зациклування. Наведемо тепер приклад, який стверджує існування неперелічених множин і проблем, які не є частково розв'язними. Проблема зациклування полягає в тому, щоб з'ясувати: чи існує алгоритм (хоча б частковий), який визначає наперед для всякої машини Тьюрінга і довільного слова p , буде чи ні дана машина працювати над словом p нескінченно довго, тобто необхідно встановити, буде чи ні розв'язною або переліченою множиною $M_c = F(X)^2$ всіх пар слів, таких, що перше слово — словникове представлення машини Тьюрінга, а друге — слово, на якому ця машина зациклюється.

Теорема 7.1.2. *Проблема зациклування машин Тьюрінга не є частково розв'язуваною.*

Доведення цієї теореми опускається.

Контрольні питання

1. Дайте означення машини Тьюрінга.
2. Яка різниця між машинами Поста і машинами Тьюрінга?
3. Яка функція називається характеристичною функцією множини $M \subseteq F(X)$?
4. Дайте визначення часткової характеристичної функції.
5. Дайте визначення множини, яка розв'язна і перелічена.

Задачі і вправи

1. Побудуйте машину Тьюрінга, що виконує: а) додавання; б) множення; в) ділення; г) обчислює модуль різниці натуральних чисел.
2. Побудуйте машину Тьюрінга, що моделює роботу заданої машини Поста.
3. Побудуйте машину Поста, що моделює роботу заданої машини Тьюрінга.
4. Побудуйте машину Тьюрінга, яка виконує конкатенацію двох слів із $F(X)$.
5. Покажіть, що проблема зупинки алгоритмічно розв'язна для машин Тьюрінга, програма яких складається з однієї команди.
6. Побудуйте машину Тьюрінга, яка витирає зі стрічки довільне початкове слово і записує на стрічку слово 11001 в алфавіті $\{0, 1\}$.

§ 7.2. СКІНЧЕННІ АВТОМАТИ З ОДНІЄЮ СТРІЧКОЮ

1. ОЗНАЧЕННЯ АВТОМАТА З ОДНІЄЮ СТРІЧКОЮ

Машини Тьюрінга виникли як модель обчислень у зв'язку з необхідністю вивчення самого поняття обчисленності і встановлення взаємозв'язків з основами математики. При розв'язуванні практичних задач, що пов'язані насамперед з розробкою ЕОМ, не завжди виникає потреба в застосуванні найзагальнішої моделі алгоритмів, а можна обйтися менш загальними і “більш розв'язними” моделями. Тому були впроваджені менш загальні моделі, ніж машини Тьюрінга, розв'язки задач для яких давали хороші практичні результати. Одна з таких моделей — скінченні automati з однією стрічкою.

Означення 7.2.1. Скінченним одностороннім детермінованим автомatem з однією стрічкою над алфавітом X називається система $A = (A, X, F, a_0, \#, I)$, де:

A — скінчена непуста множина станів;

X — алфавіт ($A \cap X = \emptyset$);

$F \subseteq A$ — множина заключних (виділених) станів;

a_0 — виділений початковий стан;

$\#$ — пустий символ;

I — програма автомата, що являє собою множину команд, де команда — це слово, що має вигляд $ax \rightarrow b$, де $a, b \in A$, а $x \in X$, і для всякої пари (a, x) існує єдина команда, що починається цими символами.

Неформально скінченні automati з однією стрічкою можна розглядати як спеціальну машину Тьюрінга, яка має свою специфіку, а саме:

а) виділені заключні стани;

б) машина читає символи, записані на стрічці, але нічого не може записувати на неї;

в) на кожному кроці машина, прочитавши символ із стрічки і перейшовши в новий стан згідно з програмою роботи, обов'язково переміщується вправо на одну клітинку;

г) машина зупиняється тоді і тільки тоді, коли вона досягла кінця слова, тобто символу $\#$.

Зауважимо, що програму I автомата з однією стрічкою можна розглядати як функцію переходів звичайного скінченного автомата $f: A \times X \rightarrow A$. Позначимо через $f(a_0, p) \in F$ той факт, що автомат A , почавши роботу в початковому стані a_0 зі словом p на стрічці, завершує її в одному із заключних станів.

Означення 7.2.2. Говорять, що автомат A допускає слово p , якщо $f(a, p) \in F$. Множину всіх слів із $F(X)$, які допускає автомат A , позначимо $T(A)$.

В розд. 6 ми бачили, що проблема еквівалентності для скінченних автоматів алгоритмічно розв'язна. Розглянемо ще деякі корисні властивості скінченних автоматів з однією стрічкою.

2. ПРОБЛЕМА ПУСТОТИ

Припустимо, що є деякий автомат $A = (A, X, f, F, a_0)$, про який нічого невідомо. Скільки треба перевірити слів за допомогою автомата A , щоб переконатися що $T(A) = \emptyset$.

Теорема 7.2.1. Нехай A — скінчений автомат з однією стрічкою. У цьому випадку $T(A) \neq \emptyset$ тоді і тільки тоді, коли A допускає деяке слово p довжини $l(p) < |A|$.

Д о в е д е н н я. Припустимо, що $T(A) \neq \emptyset$ і p — слово із $T(A)$ мінімальної довжини. Нехай $l(p) > |A| = n$. Звідси зразу випливає, що в графі переходів автомата A шлях із a_0 в $a \in F$ має цикл. Але тоді існує і простий шлях (див. розд. 4, § 4.3) із a_0 в a . Отже, в автоматі A із a_0 в a можна перейти за допомогою деякого слова, довжина якого менша довжини слова p . А це суперечить тому, що p — слово мінімальної довжини. Таким чином, $l(p) < |A|$.

Теорема доведена.

Наслідок 7.2.1. Для будь-якого скінченного автомата A з однією стрічкою проблема пустоти множини $T(A)$ розв'язна.

Дійсно, для розв'язку цієї проблеми для автомата A з n станами необхідно переглянути $|X|^{n-1}$ слів довжини $n - 1$.

3. ПРОБЛЕМА НЕСКІНЧЕННОСТІ

Розглянемо тепер питання про те, коли множина $T(A)$ нескінчена.

Лема 7.2.1. Нехай A — автомат з n станами і p — слово довжини r із $T(A)$. Якщо $n \leq r$, то існують такі слова y, z, w із $F(X)$, що $x = yzw$, $z \neq e$, і всі слова, які мають вигляд $yz^m w$, належать множині $T(A)$, $m = 0, 1, 2, \dots$

Д о в е д е н н я. Як і при доведенні теореми 7.2.1, повинні існувати цілі числа k і l , $k < l \leq r$, такі, що $f(a_0, p_{0k}) = f(a_0, p_{0l})$.

Нехай $g = p_{0k}$, $z = p_{kl}$, $w = p_{lr}$. Оскільки $k < l$, то ясно, що $z \neq e$. Зрозуміло, що $x = yzw$ і $yz = p_{0l}$. Отже, $f(a_0, y) = f(a_0, yz)$.

Звідси за індукцією випливає, що $f(a_0, y) = f(a_0, yz^n)$. Тепер маємо:

$$f(a_0, p) = f(a_0, yzw) = f(f(a_0, yz)w) = f(a_0, yz^n), w) = f(a_0, yz^n w).$$

Таким чином, всі стрічки $yz^n w$ також належать множині $T(A)$.

Теорема 7.2.2. *Нехай A — автомат з n станами. Множина $T(A)$ нескінчена тоді і тільки тоді, коли $T(A)$ включає слово p довжини r , де $n \leq r \leq 2n$.*

Д о в е д е н н я. В один бік доведення випливає з леми 7.2.1. Припустимо, що $T(A)$ нескінчена. Тоді, оскільки алфавіт X скінчений, $T(A)$ повинна мати слова, довжина яких перевищує довільне ціле число. Нехай $p \in T(A)$ і $l(p) \geq n$. Тоді існують такі цілі числа k і l , $k < l \leq l(p)$, що $f(a_0, p_{0k}) = f(a_0, p_{0l})$. Візьмемо тепер нове слово p — мінімальне із всіх слів в множині $T(A)$, для яких існують цілі числа k і l , $k < l$, що задовольняють наведене вище рівняння. Припустимо, що l — найменше таке число, довжина якого не перевищує $l(p)$. Тепер ми не впевнені в тому, що $l(p) \geq n$. Таким чином, якщо $i < j < l$, то $f(a_0, p_{0i}) \neq f(a_0, p_{0j})$. Це доводить, що $l \leq n$, оскільки функція f повинна приймати не більше ніж n значень. Якщо $l \leq i < j \leq l(p)$, то $f(a_0, p_{0i}) \neq f(a_0, p_{0j})$, оскільки в протилежному випадку слово $p' = p_{0i}p_{0j}$ було б коротшим, ніж слово p , яке задовольняє накладені умови. Підраховуючи кількість індексів, які лежать між l і $l(p)$, бачимо, що $l(p) - l + 1 \leq n$. Додавши до обох частин число l і застосувавши попередню нерівність, знаходимо, що $l(p) + 1 \leq 2n$ або в зручнішому вигляді $l(p) < 2n$. Якщо б мало місце співвідношення $n \leq l(p)$, то все було б доведено, але можливо, що це не так. Припустимо, що $l(p) < n$. Нехай $y = p_{0k}$, $z = p_{kl}$, $w = p_{l'n}$, де $n' = l(p)$. Маємо нерівність, $z \neq l$. Нехай всі слова $yz^n w$ належать множині $T(A)$, і m — найменше ціле число, таке що $n \leq k + m(l - k) + (n' - l)$. Ясно, що $m \neq 0$, оскільки $k + (n' - l) < n' < n$. Якщо $2n \leq k + m(l - k) + (n' - l)$, то $n \leq k + (m - 1)(l - k) + (n' - l)$, тому що $l - k \leq n' < n$. Але це неможливо, тому що число m було вибране як найменше таке ціле число. Отже, $k + m(l - k) + (n' - l) < 2n$. Ліва частина нерівності являє собою довжину слова $yz^n w$. Цим доводиться, що в $T(A)$ існує деяке слово, що має потрібну довжину.

Теорема доведена.

Наслідок 7.2.2. Для всякого скінченного автомата з n станами і однією стрічкою можна через скінченне число кроків вияснити, чи є множина $T(A)$ нескінченною.

§ 7.3. МАГАЗИННІ АВТОМАТИ

1. ОЗНАЧЕННЯ МАГАЗИННОГО АВТОМАТА

Розглянемо ще одну модель, простішу, ніж машини Тьюрінга, але складнішу, ніж скінченні автомати. Вона складається з нескінченних автоматів спеціального виду. Автомати цього класу називаються *магазинними* або *push-down автоматами*. Магазинний автомат — це звичайний скінчений X - Y -автомат, який називають *керуючим*, в якого, крім вхідного і вихідного каналів, є ще канал для роботи з магазином. Магазин являє собою пам'ять, яка має назву *магазинної пам'яті*, де може зберігатися одне слово в деякому алфавіті Γ . Цей алфавіт називається *магазинним алфавітом* і, взагалі кажучи, може відрізнятися від вхідного і вихідного алфавітів автомата. Магазин зручно представляти у вигляді послідовності чарунок пам'яті, перенумерованих числами натурального ряду і розміщених по вертикалі таким чином, що перша чарунка завжди стає верхньою. Слово із n символів магазинного алфавіту Γ розміщується у верхніх n чарунках магазину, а решта заповнюється спеціальним пустим символом, відмінним від символів алфавіту Γ . Магазин може працювати в двох режимах: читання і запису. При читанні сприймається буква, що знаходиться у верхній чарунці. При цьому буква видається з цієї чарунки, а частина слова, що залишилося в магазині, зміщується на одну чарунку вверх. В режимі запису, навпаки, слово, що зберігається в магазині, зміщується вниз на одну чарунку, а в чарунку, яка при цьому звільнилася, записується відповідний символ.

Магазинним автоматом називається п'ятірка $A = (A, X, Y, \Gamma, \delta)$, де A — скінчена множина станів, X, Y, Γ — скінченні вхідний, вихідний і магазинний алфавіти, що не перетинаються між собою, а δ — функція переходів і виходів, взагалі кажучи, многозначна, яка відображає елементи декартового добутку $A \times X \times Y$ на елементи декартового добутку $A \times Y \times F(\Gamma)$.

Хоча магазинний автомат означається як скінчений, що має допоміжний канал для роботи з магазином, з точки зору абстрактної теорії автоматів станами магазинного автомата потрібно було б вважати пари (a, p) , де $a \in A$, p — слово, що записане в магазині. Оскільки множина слів в алфавіті Γ нескінчена, то і магазинні автомати — теж нескінчені. Надалі станом магазинного автомата умовимося вважати стан керуючого автомата, а слово p — станом магазину.

Функція переходів і виходів δ характеризує дії магазинного автомата при різних станах його керуючої частини, магазину і вхідних символах. Ці, в принципі неоднозначні, дії можна звести до одночасного виконання скінченного числа елементарних дій п'яти типів:

- 1) читання символу із вхідного каналу і переход у наступний стан, який є функцією прочитаного символу і поточного стану автомата;
- 2) передача символу у вихідний канал і переход у наступний стан. Вихідний символ і новий стан є функціями лише поточного стану автомата;
- 3) читання верхнього символу з магазину і переход у наступний стан, який є функцією прочитаного символу і поточного стану автомата;
- 4) запис у магазин символу і переход у новий стан. Символ, що записується, і новий стан автомата визначаються його поточним станом;
- 5) переход у новий стан, який є функцією поточного стану (так званий спонтанний переход).

При такому означенні функції переходів і виходів функціонування магазинного автомата можна задати прямокутною таблицею. Стовпчики цієї таблиці відповідають станам автомата, а рядки розбиті на три секції:

- секція читання вхідних символів, складається з рядків, які відповідають символам вхідного алфавіту X ;
- секція читання символів із магазину, складається із рядків, які відповідають символам магазинного алфавіту Γ ;
- секція запису, складається з рядка, в якому вказані ті символи вихідного або магазинного алфавіту, які передаються на запис у вихідний канал або канал магазину. Секція також включає стани, в які при цьому переходить автомат. В цьому ж рядку вказуються стани, в які автомат може перейти спонтанно.

Приклад 7.2.1

Магазинний автомат, функція переходів і виходів якого задані в табл. 7.3.1, працює таким чином. В стані 1 цей автомат або сприймає вхідний символ, або записує символ S в магазин. Якщо на його вхід надійде символ x , то автомат може залишитися в тому ж стані, або перейти в стан 4. В стані 2 автомат або читає символ із магазину, або видає символ u у вихідний канал.

Таблиця 7.3.1

Вхідні символи	Вхідні стани				
	1	2	3	4	5
x	1, 4		5		
y	5		2, 4	5	
Q		3, Q^{-1}			4, Q^{-1}
S		1, S^{-1}			1, S^{-1}
	2, S	2, u	4, Q	1, 2	3, 4, 5

Таблиця переходів і виходів магазинного автомата
 $A = (\{1, 2, 3, 4, 5\}, X = \{x, y\}, Y = \{u, v\}, \Gamma = \{Q, S\})$

При читанні із магазину його верхній символ видається (для цього символи Q і S записані у від'ємній степені в другій секції таблиці). В стані 4 автомат або виконує спонтанний перехід в один із станів: 1 чи 2, або сприймає вхідний символ. При цьому якщо на його вхід подається символ x , то перехід автомата вважається незавершеним, а стан невизначеним. Те саме буде і у випадку, коли автомат читає із магазину пустий символ. ▲

2. ПРЕДСТАВЛЕННЯ МОВ В МАГАЗИННИХ АВТОМАТАХ

Назовемо перехід магазинного автомата із стану a в стан b правильним, якщо як у початковому, так і в заключному стані цього переходу магазин автомата пустий. Іншими словами, перехід правильний, якщо всі символи, записані в процесі цього переходу в магазин, виявляються прочитаними з магазину і автомат ніколи не намагається прочитати символи з магазину, які він туди не записував. Магазинний автомат називається **налаштованим**, якщо в ньому виділені початковий стан a_0 і множина заключних станів $F \subseteq A$. Будемо говорити, що магазинний автомат сприймає вхідне слово p , якщо можливий такий правильний перехід із початкового стану в один із заключних станів, при якому прочитані по вхідному каналу символи складають слово p . Магазинний автомат породжує слово q , якщо можливий такий правильний перехід із початкового стану в один із заключних станів, при якому символи, подані на вихідний канал, складають слово q . Будемо також говорити, що магазинний автомат сприймає (породжує) деяку мову L , якщо він сприймає (породжує) всі слова цієї мови і тільки їх.

Теорема 7.3.1. Якщо мова L породжується деяким магазинним автомatem A , то існує магазинний автомат B , що сприймає цю мову.

Д о в е д е н н я. Переходи з першої секції, які відповідають читанню символів вхідного алфавіту, перенесемо в третю і будемо їх інтерпретувати як переходи, що супроводжуються передачею символів у вихідний канал. І навпаки, переходи з третьої секції, які супроводжуються передачею символів у вихідний канал, перенесемо в першу секцію. При цьому, очевидно, правильний перехід із початкового стану a_0 в заключний стан b , при якому сприймається слово p і породжується слово q , також буде правильним, але при цьому буде сприйматися слово q , а породжуватись — слово p .

Теорема доведена.

Це просте твердження дає можливість надалі обмежуватися магазинними автоматами без виходів, які можуть тільки сприймати мови.

3. АНАЛІЗ МАГАЗИННИХ АВТОМАТІВ

Алгоритм аналізу магазинних автоматів буде за таблицею переходів магазинного автомата систему рівнянь, найменшим розв'язком якої є мова, що сприймається цим автомatom.

Позначимо через X_{ij} мову, що сприймається деяким магазинним автомatom, якщо стан i вибрati за початковий, а стан j — як єдиний заключний стан. Для позначення множини станів, в які можна перейти за один такт роботи із стану i внаслідок елементарних переходів кожного з чотирьох типів, про які йшла мова вище (перехід, при якому виконується подача символу на вихідний канал, не розглядається), введемо такі позначення:

N_i — для спонтанних переходів;

$K_{i,x}$ — для переходів, при яких читається вхідний символ x ;

$K_{i,R}$ — для переходів, при яких читається символ R із магазину;

$M_{i,R}$ — для переходів, при яких виконується запис символу R в магазин.

Позначимо A_R множину всіх станів, в яких можливе читання символу R із магазину, ε_{ij} — подiї, які складаються із одного пустого символу, якщо $i = j$, і пустi, якщо $i \neq j$. Згiдно з означеннями правильних переходів і елементарних дiй, які вiдповiдають одному такту роботи даного автомата, мова X_{ij} визначається рiвнянням

$$X_{ij} = \varepsilon_{ij} \vee \bigvee_{l \in N_i} X_{lj} \vee \bigvee_{x \in X} \bigvee_{l \in K_{i,x}} x X_{lj} \vee \bigvee_{R \in \Gamma} \bigvee_{k \in M_{i,R}} \bigvee_{l \in A_R} \bigvee_{m \in K_{i,R}} X_{kl} X_{mj}.$$

Розглядаючи систему таких рiвнянь для всiх пар (i, j) , можна одержати її найменший розв'язок, який i буде вiзнати су-

купність мов X_{ij} ($i, j \in A$). Якщо розглядати автомат A як настроєний з початковим станом 1 і множиною заключних станів F , то мова, що сприймається цим автоматом, буде виражатися співвідношенням

$$L(A) = \bigvee_{i \in F} X_{ii}.$$

Приклад 7.3.1

Нехай $A = (\{1, 2, 3, 4\}, \{x, y\}, \{R\}, f)$ — магазинний автомат без виходів, в якому 1 — початковий стан, 3 — заключний стан, а функція переходів задається табл. 7.3.2.

Таблиця 7.3.2

Вхідні символи	Вхідні стани			
	1	2	3	4
x	2		4	
y	4			
R				3, R^{-1}
		1, R		

З таблиці переходів цього автомата легко визначити, що $N_1 = N_2 = N_3 = N_4 = \emptyset$, $K_{1,x} = \{2\}$, $K_{3,x} = K_{1,y} = \{4\}$, $K_{2,x} = K_{2,y} = K_{3,y} = K_{4,x} = K_{4,y} = \emptyset$, $K_{1,R} = K_{2,R} = K_{3,R} = \emptyset$, $K_{4,R} = \{3\}$, $M_{1,R} = M_{3,R} = M_{4,R} = \emptyset$, $M_{2,R} = \{1\}$.

Шукана система рівнянь, яка визначає мову, представлена в цьому автоматі, має такий вигляд:

$$\begin{aligned} X_{11} &= e \vee xX_{21} \vee yX_{31}; & X_{31} &= xX_{41}; \\ X_{12} &= xX_{22} \vee yX_{32}; & X_{32} &= xX_{42}; \\ X_{13} &= xX_{23} \vee yX_{33}; & X_{33} &= e \vee xX_{43}; \\ X_{14} &= xX_{24} \vee yX_{34}; & X_{34} &= xX_{44}; \\ X_{21} &= X_{14}X_{31}; & X_{41} &= \emptyset; \\ X_{22} &= e \vee xX_{14}X_{32}; & X_{42} &= \emptyset; \\ X_{23} &= X_{14}X_{33}; & X_{43} &= \emptyset; \\ X_{24} &= X_{14}X_{34}; & X_{44} &= e. \end{aligned}$$

Користуючись загальною рекурсивною схемою розв'язання таких систем рівнянь, приходимо в результаті до мови X_{13} , яка нас цікавить:

$$\begin{aligned} X_{13} &= xX_{14} \vee y; \\ X_{14} &= xX_{14}x \vee yx. \end{aligned}$$

З другого рівняння випливає, що $X_{14} = \{x^n y x^n \mid n \geq 0\}$; тоді $X_{13} = \{x^{n+1} y x^{n+1} \mid n \geq 0\} \cup \{y\} = \{x^n y x^n \mid n \geq 0\}$. ▲

4. СИНТЕЗ МАГАЗИННИХ АВТОМАТІВ

Наведемо алгоритм синтезу, який по КВ-граматиці $G = (\Sigma, T, P, \sigma)$ (див. розд. 8.2) будує магазинний автомат $= (A, X, \Gamma, a_0, a_{\text{закл}})$ з початковим станом a_0 і єдиним заключним станом $a_{\text{закл}}$. Автомат A_G сприймає мову $L(G)$, що породжується граматикою G .

Вхідний алфавіт X автомата A_G збігається з термінальним алфавітом T граматики G . Магазинний алфавіт Γ складається із символів нетермінального алфавіту і символів, які є двійниками термінальних: $\Gamma = \Sigma \cup \{\gamma_t \mid t \in T\}$. Двійники введенні для того, щоб виконувалась умова $\Gamma \cap X = \emptyset$. Множину A станів автомата опишемо в процесі задання функції переходів f .

В початковому стані a_0 автомат записує символ σ в магазин і переходить в стан $a_{\text{закл}}$ — єдиний заключний стан, знаходчись в якому автомат може читати символи з магазину.

Коли автомат A_G читає з магазину символ γ_t (двійник термінального символу), то при цьому він переходить у стан, який позначається a_t . Якщо після цього на вхід автомата A_G подається символ t , то він переходить у стан $a_{\text{закл}}$, а в решті випадків переход із стану a_t невизначений. Зазначимо, що множина $\{a_t \mid t \in \Sigma\}$ — це всі стани автомата, в яких він може читати символи з вхідного каналу.

Якщо в стані $a_{\text{закл}}$ автомат читає з магазину символ $\xi \in \Sigma$, то він переходить в стан a_ξ . Із цього стану автомат може спонтанно перейти в один із станів $a_{(p_1)}, a_{(p_2)}, \dots, a_{(p_k)}$, де p_1, \dots, p_k — всі ті і тільки ті слова в алфавіті $\Sigma \cup T$, для яких у граматиці G є продукції $\xi \rightarrow p_\epsilon$. Якщо G містить продукцію $\xi \rightarrow \epsilon$, то за стан $a_{(\epsilon)}$ вибирається стан $a_{\text{закл}}$. Із стану $a_{(p)}$ ($p \in \{p_1, \dots, p_k\}$) автомат за кілька тактів роботи (число цих тактів дорівнює довжині слова p) переходить у стан $a_{\text{закл}}$. При цьому він записує в магазин слово p і перша буква цього слова буде в магазині верхньою. При переході із стану $a_{(p)}$, де $p = t_1 \dots t_m$, $t_1, \dots, t_m \in T$, в стан $a_{\text{закл}}$ як проміжні використовуються стани $a_{(t_1 \dots t_{m-1})}, a_{(t_1 \dots t_{m-2})}$ і т. д. В стані $a_{(t_1 \dots t_k)}$ автомат A_G записує в магазин символ t_k , якщо $t_k \in \Sigma$ або його двійник, якщо $t_k \in T$, і переходить в стан $a_{(t_1 \dots t_{k-1})}$, якщо $k > 1$, або $a_{\text{закл}}$, якщо $k = 1$. Таким чином, стани автомата A_G , що забезпечують запис правих частин продукцій у магазин, можуть бути представлені у вигляді $a_{(p)}$, де p — початкове підслово правої частини деякої продукції граматики G . Зазначимо, що початкові підслова різних продукцій можуть збігатися.

Ми не даємо детального доведення того, що мова $L(G)$ збігається з мовою, яка сприймається автомatem A_G . Зазначимо ли-

ше, що це доведення виконується шляхом доведення включення однієї множини в іншу, і навпаки. А доведення кожного з таких включень використовує зв'язок між послідовністю продукцій, які застосовуються, і послідовністю, в якій символи нетермінального алфавіту читаються при виведенні (в граматиці G) або сприйманні одного і того ж слова $p \in F(T)$. Виявляється, що нетермінальні символи читаються з магазину в послідовності, яка відповідає лівому виведенню слова p в граматиці G (виведення $p_0, p_1, \dots, p_l, p_0 = \sigma, p_l = p$ і p_{l+1} одержується з p_l застосуванням продукції до одного з нетермінальних символів, який називають **лівим**, якщо на кожному з l кроків виведення продукція застосовується до найлівішого нетермінального символу).

Приклад 7.3.2

Будемо синтезувати автомат, що сприймає мову $L(G)$, де $G = (\{x, y\}, \{\sigma, \xi\}, \{\sigma \rightarrow x\sigma x, \sigma \rightarrow \xi, \xi \rightarrow y\xi y, \xi \rightarrow x\}, \sigma)$.

Згідно з наведеним вище алгоритмом синтезу визначимо вхідний і магазинний алфавіти, а також множину станів автомата A_G , що ми синтезуємо: $X = \{x, y\}$, $\Gamma = \{\sigma, \xi, \gamma_x, \gamma_y\}$, $A = \{a_0, a_{\text{закл}}, a_x, a_y, a_\sigma, a_\xi, a_{(x\sigma x)}, a_{(\sigma x)}, a_{(x)}, a_{(\xi)}, a_{(y\xi y)}, a_{(y\xi)}, a_{(y)}\}$.

Тепер можна побудувати таблицю переходів автомата, яка складається з семи рядків і тринадцяти стовпчиків (табл. 7.3.3).

Таблиця 7.3.3

Вхідні символи	Вхідні стани												
	a_0	$a_{\text{закл}}$	a_x	a_y	a_σ	a_ξ	$a_{(x\sigma x)}$	$a_{(\sigma x)}$	$a_{(x)}$	$a_{(\xi)}$	$a_{(y\xi y)}$	$a_{(y\xi)}$	$a_{(y)}$
x			$a_{\text{закл}}$										
y					$a_{\text{закл}}$								
σ		a_σ											
ξ			a_ξ										
γ_x			a_x										
γ_y			a_y										
	σ				$a_{(x\sigma x)}$	$a_{(y\xi y)}$	γ_x	σ	γ_x	ξ	γ_y	ξ	γ_y
	$a_{\text{закл}}$				$a_{(\xi)}$	$a_{(x)}$	$a_{(\sigma x)}$	$a_{(x)}$	$a_{\text{закл}}$	$a_{\text{закл}}$	$a_{(\xi)}$	$a_{(y)}$	$a_{\text{закл}}$

Слід зазначити, що описаний алгоритм не найкращий, оскільки побудовані за ним автомати не завжди прості. Так, у випадку праволінійних чи лівлінійних граматик мови, що породжуються цими граматиками, будуть регулярними, отже, можуть сприйматися скінченими автоматами. Описаний же алгоритм синтезу будує магазинний автомат, в якому магазин використовується завжди. В цьому і полягає один з основних недоліків наведеного алгоритма синтезу. Способи підвищення ефективності алгоритма синтезу можна знайти в [55].

Контрольні питання

1. Дайте означення магазинної пам'яті.
2. Дайте означення магазинного автомата.
3. Чому магазинні автомати можна розглядати як нескінченні автомати?
4. Які елементарні дії виконуються одночасно при роботі магазинного автомата?

Задачі і вправи

1. Використовуючи алгоритм аналізу магазинних автоматів побудувати мови, які:

- a) представлені в таких автоматах, як в табл. 7.3.4;
- б) що одержані в результаті виконання алгоритму синтезу у вправі 2 а)–г).

Таблиця 7.3.4

Вхідні символи	Вхідні стани				
	1	2	3	4	5
<i>a</i>	2		5		
<i>b</i>	3		4		
<i>P</i>				2, P^{-1}	
<i>Q</i>				1, Q^{-1}	
		1, <i>P</i>			4, <i>Q</i>

2. Використовуючи алгоритм синтезу магазинних автоматів, побудувати магазинні автомати, що представляють мови, задані такими граматиками:

- a) $G = (\{a, b\}, \{\sigma, \xi\}, \sigma, P)$, де

$$P = \begin{cases} \sigma \rightarrow \sigma\xi, \\ \xi \rightarrow \xi\xi, \\ \sigma \rightarrow a, \\ \xi \rightarrow b; \end{cases}$$

- б) $G = (\{a, b\}, \{\sigma, \xi\}, \sigma, P)$, де

$$P = \begin{cases} \sigma \rightarrow e, \\ \xi \rightarrow a\xi b, \\ \sigma \rightarrow a\xi b, \\ \xi \rightarrow ab, \\ \sigma \rightarrow ab; \end{cases}$$

- в) $G = (\{x, y\}, \{\sigma, \xi\}, \sigma, P)$, де

$$P = \begin{cases} \sigma \rightarrow a\sigma, \\ \sigma \rightarrow a, \\ \sigma \rightarrow b, \\ \xi \rightarrow a, \\ x \rightarrow b; \end{cases}$$

г) $G = (\{a, b\}, \{\sigma, \xi\}, \sigma, P)$, де

$$P = \begin{cases} \sigma \rightarrow a\xi, \\ \sigma \rightarrow b\xi, \\ \xi \rightarrow a\xi, \\ \xi \rightarrow b\xi, \\ \xi \rightarrow a, \\ \xi \rightarrow b. \end{cases}$$

§ 7.4. ДИСКРЕТНІ ДИНАМІЧНІ СИСТЕМИ

Поняття дискретної динамічної системи є узагальненням поняття автомата, введеним як математична модель замкнутих систем, тобто таких систем, в яких зміна станів проходить незалежно від станів навколошнього середовища.

Нехай S — деяка множина, яку називають *простором станів*. *Процесом* в просторі станів S називається будь-яке скінченне (або нескінченне) слово (надслово) в алфавіті S . *Тривалістю* скінченного процесу $p = s_0s_1\dots s_n$ називається число n , на одиницю менше довжини слова p . Зокрема, тривалість процесу $p = s$ буде вважатися нульовою. Для заданого скінченного процесу $p = s_0s_1\dots s_n$ будемо говорити, що він починається в стані s_0 , закінчується в стані s_n і записується за допомогою виразу $s_0 \xrightarrow{p} s_n$. Якщо $p = s_0s_1\dots s_n$ і $t \in \{1, 2, \dots, n - 1\}$, то говорять, що в момент часу t процес p проходить через стан s_t , який позначається $p(t)$. Умовимося позначати множину всіх процесів у просторі S через $\text{Proc}(S)$. На цій множині визначена часткова операція $r = p * q$ послідовної композиції процесів p і q : якщо p — скінчений процес, що закінчується в тому ж стані, в якому починається процес q (можливо і нескінчений), то r є словом (або надсловом), яке виникає внаслідок операції конкатенації слова p без останнього символу і слова (або надслова) q . Якщо процес r представити у вигляді послідовної композиції двох процесів: p і q ($r = p * q$), то p називають початком процесу r , q — його закінченням, а r — продовженням процесу p .

Простір станів S , який розглядається разом з деякою множиною F допустимих процесів ($F \subset \text{Proc}(S)$), називається **дискретною динамічною системою**, якщо F задовільняє такі умови замкнутості:

- всякий початок допустимого процесу допустимий;
- якщо будь-який скінчений початок нескінченного процесу p допустимий, то і сам процес p теж допустимий.

Дискретну динамічну систему з простором станів S і множиною F допустимих процесів будемо позначати (S, F) . Прикладами дискретних динамічних систем можуть бути як детерміновані, так і недетерміновані автономні автомати без виходів. У цьому випадку S являє собою множину станів автомата, а множина F визначається початковим станом і його функцією переходів.

Для системи (S, F) визначимо функцію переходів δ як відображення, при якому кожному процесу p із F ставиться у відповідність множина $\delta(p) = \{s \mid ps \in F\}$ всіх тих станів, якими можна продовжити цей процес. Таким чином, функція переходів у системі однозначно визначається множиною допустимих процесів. Легко зрозуміти, що коли відома множина допустимих процесів нульової тривалості, то функція переходів дає можливість визнати і всю множину допустимих процесів.

Виходячи з означення функції переходів дискретної динамічної системи, можна виділити підкласи, які становлять інтерес. Система називається **детермінованою**, якщо для кожного скінченного допустимого процесу p множина $\delta(p)$ включає не більше одного стану. Для детермінованих систем заданням початкового процесу визначається весь процес. Система називається **автоматною**, якщо для будь-якого скінченного допустимого процесу $s_0s_1\dots s_n$ функція переходів задовільняє співвідношення $\delta(s_0s_1\dots s_n) = \delta(s_n)$, тобто процес, який передує попаданню в стан s_n , не впливає на подальший перехід. Система називається **вільною**, якщо вона автоматна і для всіх s із S виконується рівність $\delta(s) = S$ і множина допустимих процесів нульової тривалості збігається з S . Для вільної системи, як легко зрозуміти, множина допустимих процесів збігається з множиною $\text{Proc}(S)$.

Система S з множиною допустимих процесів F називається **налаштованою**, якщо в ній виділені такі множини S_0 і S^* початкових і заключчих станів, що $S_0 \cap S^* = \emptyset$. Пара (S_0, S^*) називається **налаштуванням системи**. В системі, де задане налаштування, виділяють допустимі процеси, що починаються в станах із S_0 і завершуються в станах із S^* . Такі процеси називаються **термінальними**. Множина всіх таких процесів позначається $\text{Term}(S)$. Інколи при означенні термінальних процесів вимагають також, щоб вони не містили заключчні стани як проміжні.

Розглянемо приклади дискретних динамічних систем, що являють собою композиції автоматів.

Дискретні перетворювачі. Нехай A — скінчений X - Y -автомат Мілі; B — Y - X -автомат Мура (як правило, нескінчений). Простором станів дискретної динамічної системи є множина всіх пар (a, b) , де $a \in A$, $b \in B$. Функцією переходів цієї системи задамо рівністю

$$\delta(a, b) = (a', b'),$$

де $a' = \delta_A(a, x)$ і x — позначка стану b в автоматі Мура B ; $b' = \delta_B(b, y)$ і $y = \lambda_A(a, x)$. Як правило, дискретні перетворювачі є налаштованими, причому налаштування (S_0, S^*) має вигляд $S_0 = (a, b)$, $S^* = (A^*, B)$, $a \in A$, $b \in B$, $A^* \subset A$, тобто функціонування схеми починається з установки обох автомата, що входять в систему, в деякі фіксовані стани, а закінчується в той момент, коли автомат A , що керує системою, опиниться в одному із заключних станів. Стан автомата B , в якому він перебуває в момент зупинки автомата A , трактується як результат роботи дискретного перетворювача. Автомат A в такій системі називається **керуючим**, а автомат B — **операційним**. Графічно дискретний перетворювач можна зобразити у вигляді такої композиції автомата A і B , в якій вхід одного автомата з'єднаний з виходом іншого, і навпаки (рис. 7.4.1).

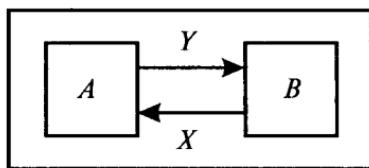


Рис. 7.4.1

Машини Тьюрінга. Машина Тьюрінга теж може бути представлена у вигляді композиції двох автомата. Один з них — скінчений автомат Мілі — називається **керуючим**, а другий — нескінчений автомат Мура — **стрічкою**. Вхідним алфавітом автомата Мілі і вихідним алфавітом автомата Мура є алфавіт X машини Тьюрінга. Станами стрічки є пари (g, n) , де $g: Z \rightarrow X$ — відображення множини цілих чисел в алфавіт X , а n — номер клітинки стрічки, на яку встановлена головка машини Тьюрінга. Настройка машини Тьюрінга виконується так, що керуючий автомат встановлюється в деякий початковий стан, фіксується початкове заповнення стрічки і головка машини встановлюється на клітинку з номером 0.

Зазначимо, нарешті, що магазинні автомати теж можна розглядати як дискретні динамічні системи.

Контрольні питання

1. Дайте означення дискретної динамічної системи.
2. Наведіть відомі Вам приклади дискретних динамічних систем.
3. Дайте означення дискретного перетворювача.
4. Чи буде скінчений автомат з однією стрічкою дискретною динамічною системою?

Задачі і вправи

1. Дайте повну і строгу модель машини Тьюрінга як дискретної динамічної системи (більш детальну, ніж наведена вище).
2. Представте машину Тьюрінга у вигляді дискретного перетворювача.
3. Представте дискретний перетворювач машиною Тьюрінга, якщо це можливо.

§ 7.5. U-Y-СХЕМИ ПРОГРАМ НАД ПАМ'ЯТЮ

Розглянемо одну з найпоширеніших графових моделей програм — *U-Y-схему* програми над пам'ятю.

Нехай D — деяка множина, на якій визначені операції сигнатури Ω , предикати сигнатури Π і в якій набувають свої значення змінні з множини $R = \{r_1, r_2, \dots, r_m\}$ (пам'ять). Пара (D, Ω) — це універсальна Ω -алгебра, яку будемо називати *алгеброю даних*, а часткове відображення $b : R \rightarrow D$ — *станом пам'яті*. Множину станів пам'яті B будемо називати *інформаційним середовищем*.

Якщо $T(\Omega, R)$, як і раніше, означає Ω -алгебру термів над R , то відображення b можна продовжити на всю множину $T(\Omega, R)$, коли покласти

$$b(t(r_1, \dots, r_m)) = t(b(r_1), \dots, b(r_m)).$$

При цьому якщо $b(r_i)$ ($i = 1, 2, \dots, m$) не визначено, то ліва і права частини вважаються невизначеними, а рівність двох виразів має місце тоді і тільки тоді, коли обидві частини набувають однакові значення або обидві невизначені.

Розглянемо множину виразів виду $u(t_1, \dots, t_n)$, де u — символ n -арного предиката із Π , а $t_1, \dots, t_n \in T(\Omega, R)$. Кожний такий ви-

раз означає предикат на множині B . Значення цього предиката при заданому стані пам'яті $b \in B$ визначається як

$$u(t_1, \dots, t_n)(b) = u(b(t_1), \dots, b(t_n)).$$

Нехай $U(R, \Omega, \Pi)$ — множина всіх таких виразів, а \hat{U} — множина пропозиційних булевих функцій від виразів із $U(R, \Omega, \Pi)$. Елементи множини $U(R, \Omega, \Pi)$ будемо називати **базовими умовами над пам'яттю** R , а елементи множини \hat{U} — **елементарними умовами над пам'яттю** R .

Оператором присвоювання називається вираз, що має вигляд

$$y = (r := t(r)) = (r_1 := t_1(r), \dots, r_m := t_m(r)),$$

де $r = (r_1, \dots, r_m)$, $t_1(r), \dots, t_m(r) \in T(\Omega, R)$, $r_1, \dots, r_m \in R$. Кожний оператор присвоювання у задає деяке перетворення на множині B . Перетворення, яке виконується оператором y на B , визначається рівністю

$$y(b) = \begin{cases} b(t_i), & r_i \in \{r_1, \dots, r_m\} \ (i = 1, 2, \dots, m); \\ b(r), & r_i \notin \{r_1, \dots, r_m\}. \end{cases}$$

Якщо $g = y_1 y_2 \dots y_k$ — добуток операторів присвоювання, то йому відповідає перетворення y_g , яке дорівнює послідовній суперпозиції перетворень, що відповідають операторам y_1, y_2, \dots, y_k . Нехай $Y(R, \Omega)$ — множина всіх операторів присвоювання. Елементи цієї множини будемо називати **базовими операторами**.

Пара $(U(R, \Omega, \Pi), Y(R, \Omega))$ називається **стандартним базисом над пам'яттю** R , що визначається сигнатурою (Ω, Π) . Якщо $U \subset U(R, \Omega, \Pi)$, $Y \subset Y(R, \Omega)$, то (стандартною) **U - Y -схемою програми над пам'яттю** R (або просто **U - Y -схемою програми**) називається множина станів A схеми програми разом з множиною переходів $S \subset A \times U \times Y^* \times A$ і двома виділеними станами: **початковим** a_0 і **заключним** a^* , де Y^* — множина всіх слів в алфавіті Y . Слови в алфавіті Y , тобто послідовності базових операторів, будемо називати **елементарними операторами**.

Нехай $(a, u, y, b), (d, u_1, y_1, c) \in S$. Якщо $a = d$, а $b \neq c$ або $y \neq y_1$, то стан a називають **роздгалуженням**; якщо ж $b = c$, а $a \neq d$ або $y \neq y_1$, то стан b називають **злиттям**.

Якщо $(a, u, y, b) \in S$, то говорять, що даний переход веде із стану a в стан b . Виділені стани характеризуються тим, що в S немає жодного переходу, який веде в стан a_0 , і жодного переходу, який веде із a^* . **Шляхом l довжини n** , що веде із стану a в стан b ($l(a, b)$), називається послідовність станів $a_1 = a, a_2, \dots, a_n = b$, зв'язаних переходами (a_i, u_i, y_i, a_{i+1}) , $i = 1, \dots, n - 1$. При цьому шлях l із стану a в стан b називається **лінійною ланкою**, якщо для

всіх $i = 1, \dots, n - 1$ в множині переходів S існує єдиний переход, який веде із стану a_i в a_{i+1} . Стан a називають **входом** лінійної ланки, а стан b — **виходом**.

З кожною U - Y -схемою програми над пам'яттю можна зв'язати орграф $G = (A, E)$, дуги якого помічені таким чином:

якщо $(a, u, y, b) \in S$, то $(a, b) \in E$ і помічено парою u/y , а $\text{Pr}(a_0) = \emptyset$ і $\text{Ps}(a^*) = \emptyset$.

Процес виконання U - Y -схеми програми при заданому початковому стані пам'яті $b_0 \in B$ — це послідовність пар $(a_1, b_1), (a_2, b_2), \dots, (a_n, b_n)$, ..., де $a_1 = a_0, b_1 = b_0, a_i \in A, b_i \in B (i = 1, 2, \dots)$, а для всякої пари (a_i, a_{i+1}) станів U - Y -схеми програми в множині S існує переход (a_i, u_i, y_i, a_{i+1}) , такий, що $u_i(b_i) = 1$ і $b_{i+1} = y_i(b_i)$. Процес виконання U - Y -схеми програми називається **термінальним**, якщо він скінчений і остання пара має вигляд (a^*, b) , де a^* — заключний стан.

U - Y -схема програми разом із заданим початковим станом пам'яті $b_0 \in B$ буде називатися **U - Y -програмою над пам'яттю** (або просто **U - Y -програмою**).

U - Y -програми задаються різними способами. Найбільш поширеними є такі три представлення: текстове, графове і у вигляді складеного об'єкта.

Приклад 7.5.1

U - Y -програму множення цілих додатних чисел x і y , яку будемо позначати $\text{УМН}(x, y)$ з результатом z , можна задати трьома способами:

початок.

$u := x;$

$z := 0;$

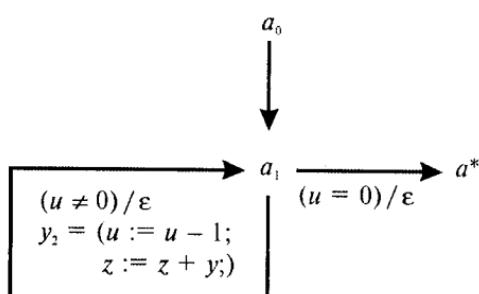
поки ($u \neq 0$) виконувати

$u := u - 1;$

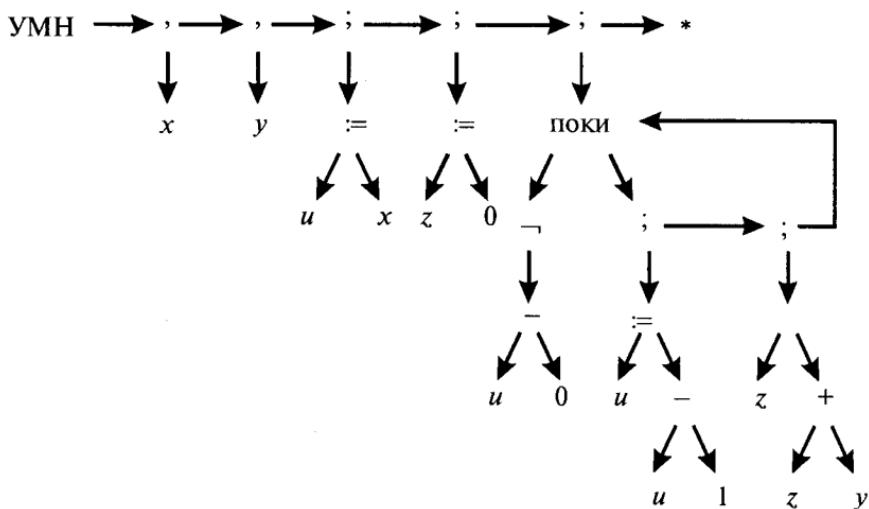
$z := z + y;$

кінець.

Текстове представлення
 U - Y -програми $\text{УМН}(x, y)$



Графове представлення
 U - Y -програми $\text{УМН}(x, y)$



Представлення U - Y -програми $\text{УМН}(x, y)$ складеним об'єктом
(символ $*$ означає пустий складений об'єкт).

Схеми програм над пам'яттю можна представляти за допомогою термів алгебри алгоритмів Глушкова Z . Дійсно, якщо за базис множини ОП взяти множину операторів присвоювання, а за базис для УМ — множину елементарних умов, то між алгеброю Z і U - Y -схемами програм над пам'яттю встановлюється зв'язок, який виражається такою теоремою.

Теорема 7.5.1. Всякому регулярному виразу із Z можна поставити у відповідність U - Y -схему програми над пам'яттю, і навпаки, всякій U - Y -схемі програми над пам'яттю можна поставити у відповідність регулярний вираз алгебри алгоритмів Z [15].

U - Y -схему програми, побудовану за регулярним виразом, будемо називати *регулярною*.

Приклад 7.5.2

Представлення U - Y -програми $\text{УМН}(x, y)$ задано регулярним виразом:

$$t = (u := x; z := 0;) \{u := u - 1; z := z + y;\} = PQR \{ST\}.$$

Як уже зазначалось в розділі 2, можливість представлення програми у вигляді виразу алгебри алгоритмів дозволяє проводити над програмами формальні перетворення. Ці перетворення ґрунтуються насамперед на *тотожніх і квазитотожніх співвідношеннях* алгебри алгоритмів, а також на *інваріантних* співвідношеннях, про які мова піде дещо пізніше.

Контрольні питання

1. Дайте означення *U-Y*-схеми програми над пам'яттю.
2. Які операції алгебри алгоритмів Ви знаєте?
3. Яка різниця між тотожністю і квазітотожністю?
4. Яка *U-Y*-програма називається регулярною?
5. Яким чином позначаються вершини і дуги в графі переходів *U-Y*-програми?

Задачі і вправи

1. Дайте доведення теореми 7.5.1.
2. Чи можна представити схему програми над пам'яттю за допомогою машини Тьюрінга?
3. Яка програма відповідає схемі програми над пам'яттю, граф якої ациклічний.

Р о з д і л 8

ФОРМАЛЬНІ ГРАМАТИКИ І ФОРМАЛЬНІ МОВИ

Появі поняття формальної мови завдячують насамперед пошуку підходящої математичної моделі для природних мов, таких, як українська, російська, англійська та ін. В 1959—60-х р. з'явилося кілька робіт, що заклали основи формальних мов і формальних граматик. Значний прогрес у цьому напрямку почався в 1960 р., коли було виявлено, що клас формальних мов типу АЛГОЛ-60 збігається з класом контекстно-вільних мов. Велику участь у цьому процесі взяли спеціалісти по застосуванню ЕОМ для дослідження природних мов і спеціалісти з мов програмування. При цьому було одержано ряд теоретичних результатів, пов'язаних з математичним забезпеченням ЕОМ і, зокрема, з методами програмування.

У даному розділі розглядаються основні поняття і методи теорії формальних граматик і мов, які мають назву *регулярних* і *контекстно-вільних*.

§ 8.1. РЕГУЛЯРНІ ГРАМАТИКИ ТА ЇХ ВЛАСТИВОСТІ

1. ОЗНАЧЕННЯ ФОРМАЛЬНОЇ ГРАМАТИКИ

Всяка формальна мова — це множина слів у деякому скінченному алфавіті. До таких мов належать, зокрема, штучні мови для спілкування людини з обчислювальною системою, які називаються *мовами програмування*. Для того щоб задати таку штучну мову, необхідно:

— задати алфавіт, тобто множину символів (або букв), кожен з яких можна розмножувати необмеженою кількістю екземплярів;

— описати формальну граматику мови, тобто множину правил, за якими із символів алфавіту будуються такі послідовності символів, що належать даній мові і називаються **правильними словами**. Слова відокремлюються одне від одного за допомогою спеціального символу алфавіту — **пропуску** (проміжку стандартної довжини) між символами алфавіту, який перевищує довжину будь-якого з проміжків, що, можливо, зустрічаються всередині символів алфавіту.

Необхідно зауважити, що правила формальної граматики можна розглядати як **правила виведення** — елементарні операції, які застосовуються в деякій послідовності до початкового слова і породжують лише правильні слова. Сама послідовність правил, які використовуються в процесі породження даного слова, називається **виведенням**.

Формальні граматики за способом задання правильних слів поділяються на **граматики, що породжують слова**, і **граматики, що розпізнають слова**. До граматик першого типу відносять граматики, які дають можливість будувати довільне правильне слово разом із його структурою і які не будують жодного неправильного слова. До граматик другого типу відносять граматики, які для довільного слова дають можливість встановити, правильне чи ні це слово, і коли воно правильне, то вияснити його будову.

Перейдемо до формального поняття граматики, яке вперше було сформульоване Хомським в 1967 р. [74].

Означення 8.1.1. Граматикою, що породжує слова (або просто граматикою), називається упорядкована четвірка об'єктів

$$G = (V_t, V_h, \sigma, P),$$

де V_t — скінчений алфавіт символів, елементи якого називаються **термінальними символами** (**основний термінальний алфавіт**); V_h — скінчений допоміжний алфавіт, елементи якого називаються **нетермінальними символами** і позначаються малими буквами грецького алфавіту; $\sigma \in V_h$ — **початковий нетермінальний символ (аксіома)**; $P = \{u_i \rightarrow v_i \mid i = 1, 2, \dots, k\}$ — скінчена система підстановок, ліві і праві частини яких є символами в алфавіті $V = V_t \cup V_h$. Символ \rightarrow не належить алфавіту V .

Символи алфавіту V_t являють собою елементарні одиниці мови, що породжується. Символи алфавіту V_h — метазмінні, які використовуються при виведенні правильних слів (для природних мов такими змінними є граматичні класи: іменники, дієслова, займенники і т. д.). Символ σ — аксіома, з якої виводяться всі правильні слова (в природних мовах аксіомі відповідає граматичний клас “речення”), а P — схема граматики. Вона складається з правил виведення, тобто граматичних правил мови, що породжуються.

Приклад 8.1.1

Нехай $G_0 = (V_T = \{a, b\}, V_H = \{\sigma, \tau\}, \sigma, P = \{\sigma \rightarrow a\sigma a, \sigma \rightarrow c, \sigma \rightarrow \tau\tau, a\sigma a \rightarrow b\})$. Граматика G_0 є граматикою, що породжує слова в алфавіті $V = \{a, b, \sigma, \tau\}$.

Нехай $x, y \in F(V)$ — довільні слова вільної напівгрупи слів над алфавітом $V = V_T \cup V_H$.

Слово y називається *словом, яке безпосередньо виводиться* із слова x , якщо в граматиці G знайдеться таке правило $(u \rightarrow v) \in P$, що $x = x_1ux_2$, $y = x_1vx_2$, $x_1, x_2 \in F(V)$, і слово y можна одержати із слова x шляхом заміни слова u в слові x на слово v ($x \xrightarrow[G]{} y$).

Транзитивне замикання $\xrightarrow[G]{*}$ відношення безпосереднього виведення називається *відношенням виведення*. Іншими словами, слово y називається словом, що виводиться із слова x , якщо слова x і y збігаються або коли існує послідовність слів z_0, z_1, \dots, z_l , така, що $z_0 = x$, $z_l = y$, і для будь-якого $1 \leq i \leq l$ має місце $z_i \xrightarrow[G]{} z_{i+1}$ ($x \xrightarrow[G]{*} y$). Послідовність слів z_0, z_1, \dots, z_l називається *виведенням*.

Приклад 8.1.2

Нехай G_0 — граматика, що наведена в попередньому прикладі. Тоді в цій граматиці:

a) $b\sigma c \xrightarrow[G]{*} b\tau c \text{ і } acaba \xrightarrow[G]{} bba;$

б) $\sigma \xrightarrow[G]{*} aba$, оскільки послідовність $\sigma, a\sigma a, aa\sigma aa, aaaa\sigma aa, aba$ є виведенням слова aba із слова σ .

Необхідно підкреслити, що застосування правил граматики при побудові виведень не є “жорстким”, тобто на кожному кроці виведення можна вибирати довільне правило із P , яке застосоване в даний момент. Це означає, що порядок застосування правил у граматиці довільний і всяке правило дозволяється застосовувати після будь-якого правила. Необхідно лише, щоб дане правило було застосовним. Цим поняття формальної граматики, що породжує слова в даному алфавіті V , суттєво відрізняється від поняття нормального алгорифму Маркова (див. т.1, § 3.3), де підстановки виконуються в строго заданій послідовності.

Виведення $x \xrightarrow[G]{*} y$ називається *повним*, якщо $y \in F(V_T)$, тобто коли слово y складається лише з термінальних символів. Отже, всяке повне виведення завершується застосуванням правил, пра-

ві частини яких є словами в алфавіті V_T . Такі правила будемо називати **заключними правилами підстановки**, або **заключними підстановками** даної граматики.

Якщо $x \rightarrow^* y$ і $y \notin F(V_T)$, причому в системі P не існує правил, застосовних до слова y , то виведення y із x в граматиці G називається **тупиковим**.

Приклад 8.1.3

Наведене вище виведення слова aba в граматиці G_0 із слова σ є повним, а правила $\sigma \rightarrow c$ і $aca \rightarrow b$ — заключними підстановками, в той час як виведення σ , $a\sigma a$, $a\sigma aa$, $a\sigma a a$ є прикладом тупикового виведення.¹

Кожному способу виведення слова однозначно відповідає дерево виведення (рис. 8.1.1), яке в лінгвістиці називається **деревом синтаксичного аналізу**. Вершини цього дерева позначаються символами, що входять в слова, які породжуються в процесі виведення. Корінь дерева позначається аксіомою σ . З кожної вершини дерева, позначеної нетермінальним символом, замість якого на наступному кроці виведення при відповідному контексті вставляється деяке слово, виходять дуги, які ведуть до вершин, позначених символами слова, що вставляється. Наприклад, повне виведення слова a^3ba^3 , що утворює послідовність (σ , $a\sigma a$, $a^2\sigma a^2$, $a^3\sigma a^3$, a^3ba^3), має таке дерево виведення: слово $x \in F(V_T)$ називається **правильним у граматиці G** , якщо існує хоча б одне повне виведення x із аксіоми σ в граматиці G , тобто слово x правильне, якщо: 1) $x \in F(V_T)$; 2) $\sigma \xrightarrow{G}^* x$.

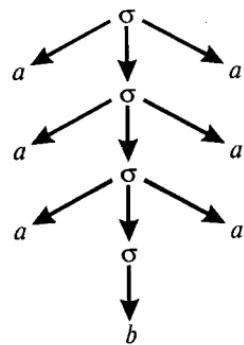


Рис 8.1.1

Множина $L(G)$ всіх правильних слів у граматиці G називається **мовою, що породжена граматикою G** .

Таким чином, кожній граматиці $G = (V_T, V_H, \sigma, P)$ однозначно відповідає мова $L(G)$, що породжується даною граматикою. Але ця відповідність не є взаємно однозначною: одна і та ж мова може бути породжена різними граматиками. Це дає підстави ввести відношення еквівалентності на множині граматик:

G і G' еквівалентні ($G \sim G'$), якщо $L(G) = L(G')$.

Приклад 8.1.4

Граматика G_0 із попередніх прикладів породжує мову

$$L(G_0) = \{a^nba^n \cup a^mca^m \mid m, n = 0, 1, 2, \dots\}.$$

Граматика $G_1 = (\{a, b, c\}, \{\sigma\}, \sigma, P)$, де

$$P = \begin{cases} \sigma \rightarrow a\sigma a, \\ \sigma \rightarrow b, \\ \sigma \rightarrow c, \end{cases}$$

породжує мову $L(G_1)$, що збігається з мовою $L(G_0)$, і тому $G_0 \sim G_1$. \blacktriangleleft

В класі еквівалентних граматик, тобто граматик, що породжують одну і ту ж мову, логічно ввести **нормальні форми** — граматики, які задовольняють певні обмеження.

Теорема 8.1.1. Для всякої граматики $G = (V_T, V_H, \sigma, P)$, що породжує слова, існує еквівалентна їй граматика $G' = (V_T, V_H', \sigma, P')$, ліва частина кожного правила якої не включає символів алфавіту V_T [19].

Д о в е д е н н я. Дійсно, нехай $G = (V_T, V_H, \sigma, P)$ — довільна граматика, що породжує слова в алфавіті $V = V_T \cup V_H$. Кожному термінальному символу $a \in V_T$ поставимо у відповідність додатковий нетермінальний символ $a' \notin V_H$, так, щоб різним символам алфавіту V_T відповідали різні нетермінальні символи. Нехай $U_H = \{a' \mid a \in V_T\}$ — одержана в результаті множина додаткових нетермінальних символів. Побудуємо граматику $G' = (V_T, V_H', \sigma, P')$, де $V_H' = V_H \cup U_H$. Множина правил P' будеться за правилами граматики G так, що в кожній продукції схеми P всі термінальні символи замінюються відповідними нетермінальними. Крім того, в систему P' для кожного $a \in V_T$ включаються правила, що мають вигляд $a' \rightarrow a$, які є заключними правилами граматики G' . Звідси випливає, що ліва частина кожного правила схеми P' не включає символів основного алфавіту V_T .

Доведемо тепер еквівалентність граматик G і G' . Якщо $x = a_{i_1}a_{i_2}\dots a_{i_r} \in L(G)$ і $W = (\sigma, z_1, \dots, z_k)$ ($x = z_k$ — деяке повне виведення слова x в граматиці G), то з даного виведення неважко побудувати повне виведення W' того ж слова x в граматиці G' . Для цього достатньо на i -му кроці виведення W' скористатися правилом $p'_i \in P'$, яке відповідає правилу p_i граматики G і було застосоване на i -му кроці виведення W ($i = 1, 2, \dots, k$). В результаті одержимо виведення

$$W'' = (\sigma, z'_1, \dots, z'_k),$$

де $z'_k = a'_{i_1}a'_{i_2}\dots a'_{i_r}$.

Застосовуючи, нарешті, заключні правила підстановки грама-

тики G' , одержимо повне виведення слова x в граматиці G' . Значить, $L(G) \subseteq L(G')$. Нехай тепер $x = a_{i_1}a_{i_2}\dots a_{i_r} \in L(G')$ і $W' = (\sigma, z_1, \dots, z_k)$ ($x = z_k$ — деяке повне виведення слова x в граматиці G). Оскільки символи основного алфавіту не входять в ліві частини правил підстановки граматики G' , то виведення W' можна передбувати так, щоб застосування правил $a' \rightarrow a$ використовувалось лише в самому кінці — після застосування всіх інших правил граматики G' . Перебудоване таким чином виведення W' може бути розбите на дві частини: перша — W'' не включає застосування заключних правил підстановки і закінчується словом $x' = a_{i_1}'a_{i_2}'\dots a_{i_k}'$, а друга — складається з послідовного застосування заключних правил граматики G' , які переводять слово x' в слово x . Якщо стерти всі штрихи в першій частині W'' , то одержимо виведення слова x в граматиці G . Отже, $L(G') \subseteq L(G)$. Таким чином, $L(G) = L(G')$ і граматики G і G' еквівалентні.

Теорема доведена.

Граматика G' , побудована в теоремі, називається *нормальнюю формою граматики G*, що породжує слова.

2. КЛАСИФІКАЦІЯ ГРАМАТИК

Відомо, що сімейство мов, що породжуються граматиками, збігається з сімейством рекурсивно перелічених множин [51, 13], тобто граматики можуть породжувати множини досить загального вигляду. Для більшої конкретизації таких множин на правила підстановки граматики накладають певні обмеження. Кожне з таких обмежень приводить до виникнення деякого спеціального сімейства формальних мов. Розглянемо декотрі з таких обмежень.

Означення 8.1.1. Граматика $G = (V_T, V_H, \sigma, P)$, яка породжує слова в алфавіті $V = V_T \cup V_H$, називається *граматикою безпосередніх складових (бс-граматикою)*, якщо кожне з її правил підстановки має такий вигляд:

$$u\xi v \rightarrow uyv,$$

де $\xi \in V_H$, $u, v, y \in F(V)$. Слова u і v називаються відповідно *лівим і правим контекстом* даного правила.

Мова $L(G)$, що породжена бс-граматикою G , називається *бс-мовою*.

Таким чином, заміна символу ξ словом u можлива лише в контексті, лівою частиною якого є слово u , а правою — слово v .

Слід зауважити, що множина всіх бс-граматик є власною підмножиною множини всіх граматик, що породжують слова в

деякому алфавіті, оскільки доведено, що не для всякої такої граматики існує еквівалентна їй бс-граматика [14].

Нехай $G = (V_T, V_H, \sigma, P)$ — граматика, що породжує слова в алфавіті $V = V_T \cup V_H$. Граматика G називається *граматикою, що не вкорочує слова*, якщо для кожного правила $u \rightarrow z$ граматики G має місце нерівність $l(u) \leq l(z)$, де $l(p)$ означає довжину слова p .

Означення 8.1.2. Граматика $G = (V_T, V_H, \sigma, P)$, що породжує слова в алфавіті $V = V_T \cup V_H$, називається *контекстно-вільною граматикою (кв-граматикою)*, якщо кожне з її правил підстановки має такий вигляд:

$$\xi \rightarrow v,$$

де $\xi \in V_H$, $v \in F(V)$.

Мова $L(G)$ називається *контекстно-вільною мовою (кв-мовою)*, якщо вона породжується деякою кв-граматикою G . Термін контекстно-вільна в назві граматики означає той факт, що заміна нетермінального символу деяким словом не залежить від контексту.

Приклад 8.1.5

1. Мова $L(G) = \{xbx \mid \forall x \neq e \in F(V_T)$, де $V_T = \{a_1, \dots, a_m\}$ є бс-мовою, що породжується бс-граматикою G , заданою такою схемою:

$$\left\{ \begin{array}{l} \delta \rightarrow \sigma \psi_i \alpha_i, \\ \delta \rightarrow \beta, \\ \alpha_i \psi_j \rightarrow \delta_{ij} \psi_j, \\ \delta_{ij} \psi_j \rightarrow \delta_j \alpha_i, \\ \delta_{ij} \alpha_i \rightarrow \psi_j \alpha_i, \\ \beta \psi_i \rightarrow \tau_i \psi_i, \\ \tau_i \psi_i \rightarrow \tau_i \beta, \\ \tau_i \beta \rightarrow \alpha_i \beta, \\ \beta \rightarrow b, \\ \alpha_i \rightarrow a_i \end{array} \right.$$

для довільних $i, j = 1, 2, \dots, m$.

2. Мова $L(G)$, що породжується граматикою $G = (\{a, b\}, \{\sigma\}, \sigma, P)$, де

$$P = \left\{ \begin{array}{l} \sigma \rightarrow a, \\ \sigma \rightarrow abab, \end{array} \right.$$

є кв-мовою. ▲

ГоворяТЬ, що кв-граматика $G = (V_T, V_H, \sigma, P)$, що породжує слова в алфавіті $V = V_T \cup V_H$, суттєво залежить від нетермінального символу ψ , якщо виконуються такі умови:

1) існує хоча б одне слово $z = u_\psi v$, яке виводиться з аксіоми σ , тобто $\sigma \xrightarrow[G]{*} z$;

2) існує хоча б одне термінальне слово $x \in F(V_T)$, яке виводиться із ψ , тобто $\psi \xrightarrow[G]{*} x$.

В силу скінченності алфавіту V_H і множини P для кожного $\psi \in V_H$ можна встановити, залежить суттєво граматика $G = (V_T, V_H, \sigma, P)$ від ψ чи ні. Символ $\psi \in V_H$ називається **фіктивним**, якщо граматика $G = (V_T, V_H, \sigma, P)$ суттєво не залежить від ψ . Ясно, що всяке правило підстановки граматики G , яке включає хоча б один фіктивний символ, може бути вилучене із системи P , а всі фіктивні символи — з алфавіту V_H .

Кв-граматика $G = (V_T, V_H, \sigma, P)$, яка суттєво залежить від кожного нетермінального символу, називається **приведеною**.

За допомогою бс- і кв-граматик задаються деякі із сучасних мов програмування, зокрема мова ПАСКАЛЬ [8]. У загальному випадку граматика довільної формальної мови задається у вигляді правил, які записуються в так званій **нормальній формі Бекуса (БНФ)**. Проілюструємо цей метод заданням множини L ідентифікаторів мови ПАСКАЛЬ:

$$\langle \text{ідентифікатор} \rangle ::= \langle \text{буква} \rangle | \langle \text{ідентифікатор} \rangle \langle \text{буква} \rangle | \langle \text{ідентифікатор} \rangle \langle \text{цифра} \rangle.$$

Ідентифікатори, як видно з цього фрагменту, являють собою різні послідовності, що складаються з букв латинського алфавіту і цифр 0, 1, 2, ..., 9 і починаються з деякої букви.

Мова L , що породжується наведеним фрагментом, є кв-граматикою $G = (\{A, B, C, \dots, X, Y, Z, 0, 1, 2, \dots, 9\}, \{\alpha, \beta, \gamma\}, \alpha, P)$, де

$$P = \left\{ \begin{array}{l} \alpha \rightarrow \alpha\beta, \\ \alpha \rightarrow \alpha\gamma, \\ \alpha \rightarrow \beta, \\ \beta \rightarrow A, \\ \beta \rightarrow B, \\ \dots \\ \beta \rightarrow Z, \\ \gamma \rightarrow 0, \\ \dots \\ \gamma \rightarrow 9. \end{array} \right.$$

3. ПРАВОЛІНІЙНІ І ЛІВОЛІНІЙНІ ГРАМАТИКИ (РЕГУЛЯРНІ ГРАМАТИКИ) І СКІНЧЕННІ АВТОМАТИ

Означення 8.1.3. Правило підстановки $\psi \rightarrow z$ називається *лінійним*, якщо $z \in F(V_T)$ або $z = uv$, де $u, v \in F(V_T)$ і $\psi, \tau \in V_H$. Іншими словами, права частина лінійного правила — це або слово в термінальному алфавіті, або слово, яке має лише один нетермінальний символ.

Означення 8.1.4. Правило підстановки $\psi \rightarrow z$ називається *праволінійним (ліволінійним)*, якщо $z \in F(V_T)$ або $z = ut (\tau u)$, де $u, v \in F(V_T)$ і $\psi, \tau \in V_H$, тобто якщо єдиний нетермінальний символ входить у праві лінійні частини правил підстановки, то він завжди крайній справа (зліва).

Граматика $G = (V_T, V_H, \sigma, P)$ називається відповідно *лінійною, праволінійною, ліволінійною*, якщо її правила підстановки лінійні, праволінійні, ліволінійні. Аналогічно **мова** $L(G)$ називається *лінійною, праволінійною, ліволінійною*, якщо граматика G є лінійною, праволінійною, ліволінійною відповідно.

З наведених означень випливає, що множина Λ всіх лінійних мов включає множину Λ_P всіх праволінійних і множину Λ_L всіх ліволінійних мов. Має місце така теорема.

Теорема 8.1.2. Кожній праволінійній граматиці G_P відповідає еквівалентна їй ліволінійна граматика G_L так, що $L(G_P) = L(G_L)$, і навпаки.

Д о в е д е н н я. Опишемо загальний метод побудови за приведеною формою праволінійної граматики $G = (V_T, V_H, \sigma, P)$ еквівалентної їй ліволінійної граматики $G' = (V_T, V_H', \sigma', P')$. Суть цієї побудови пояснимо на прикладі конкретної граматики $G = (V_T = \{a, b, c\}, V_H = \{\sigma, \psi, \tau\}, \sigma, P)$, де P складається з таких правил:

$$\begin{cases} \sigma \rightarrow a\psi, \\ \sigma \rightarrow a\tau, \\ \psi \rightarrow a\psi, \end{cases} \quad \begin{cases} \psi \rightarrow ac\tau, \\ \tau \rightarrow bt\tau, \\ \tau \rightarrow b. \end{cases}$$

Еквівалентна їй ліволінійна граматика має таку систему правил:

$$\begin{cases} \sigma' \rightarrow \tau b, \\ \tau \rightarrow \tau b, \\ \tau \rightarrow \psi ac, \end{cases} \quad \begin{cases} \psi \rightarrow \psi a, \\ \tau \rightarrow ac, \\ \psi \rightarrow a. \end{cases}$$

Обидві граматики породжують множину слів

$$L = \{a^n cb^m\} (m, n = 1, 2, \dots)$$

в алфавіті $V_T = \{a, b, c\}$.

Виберемо за аксіому граматики G' новий нетермінальний символ $\sigma' \notin V_H$. Покладемо $V'_H = (\{\sigma'\} \cup V_H) \setminus \{\sigma\}$. У нашому прикладі маємо $G' = (V_T = \{a, b, c\}, V'_H = \{\sigma', \psi, \tau\}, \sigma', P')$, де P' будується за множиною P таким чином. Виділимо спочатку в P всі заключні правила, що мають вигляд $s \rightarrow u$. Кожному з них в P' відповідає правило $\sigma' \rightarrow u$, де $u \in F(V_T)$. Після цього кожному заключному правилу з P , що має вигляд $\xi \rightarrow u$, $\xi \neq \sigma$, поставимо у відповідність ліволінійне правило $\sigma' \rightarrow \xi u$ в P' . У нашому прикладі є лише одне заключне правило $\tau \rightarrow b$, якому відповідає ліволінійне правило $\sigma' \rightarrow \tau b \in P'$. Нехай $V_1 = \{ \xi \mid \xi \neq \sigma \} = V_H$ — множина всіх нетермінальних символів (відмінних від аксіоми σ), з яких складаються ліві частини заключних правил граматики G . Виберемо множину правил

$$\{\delta \rightarrow u_1 \xi \mid \forall x \in V_1\} \subseteq P,$$

де $u_1 \in F(V_T)$. У нашому прикладі $V_1 = \{\tau\}$, так що до вибраних правил належать правила $\sigma \rightarrow a\sigma\tau$, $\psi \rightarrow a\sigma\tau$, $\tau \rightarrow b\tau$. Якщо серед вказаних правил є правило, що має вигляд $\sigma \rightarrow u_1 \xi$, то в граматиці G' їм відповідають заключні правила $\xi \rightarrow u_1$. Решті правил, що мають вигляд

$$\delta \rightarrow u_1 \xi, \tag{8.1.1}$$

де $\delta \neq \sigma$, ставляться у відповідність правила:

$$\xi \rightarrow \delta u_1$$

в множині P' . В даному прикладі правилу $\sigma \rightarrow a\sigma\tau$ відповідає правило $\tau \rightarrow ac \in P'$, яке є заключним в граматиці G' . В той же час правилам $\psi \rightarrow a\sigma\tau$ і $\tau \rightarrow b\tau$ ставляться у відповідність ліволінійні правила

$$\tau \rightarrow \psi ac \text{ i } \tau \rightarrow \tau b,$$

які належать схемі P' .

Далі побудуємо множину $V_2 = \{\delta \mid \delta \notin V_1 \cup \{\sigma\}\} \subseteq V_H$ всіх нетермінальних символів (які відмінні від σ і не входять в V_1), з яких складаються ліві частини правил (8.1.1). Виділимо сукупність правил $\{\rho \rightarrow u_2 \delta \mid \text{для всіх } \delta \in V_2\} \subseteq P$, де $u_2 \in F(V_T)$. Виконаємо для них побудову, аналогічну попередній, тобто кожному правилу, що має вигляд $\sigma \rightarrow u_2 \delta$, відповідає заключне правило $\delta \rightarrow u_2 \in P'$, а кожному правилу $\rho \rightarrow u_2 \delta$ (де $\rho \neq \sigma$) — ліволінійне правило $\delta \rightarrow \rho u_2$ із P' і т.д. У нашому прикладі $V_2 = \{\psi\}$ і правилам $\sigma \rightarrow \psi a$, $\psi \rightarrow a\psi$ в граматиці G' відповідають заключні правила $\tau \rightarrow a$ і правило $\psi \rightarrow \psi a$. Внаслідок скінченності множини V_H і того, що граматика G приведена, цей процес закінчиться на

деякому скінченному кроці побудови множини правил підстановки P' ліволінійної граматики G' . Побудована за цим методом для нашого прикладу граматика G' була наведена вище.

Таким чином, користуючись множиною правил P граматики G , можна побудувати ліволінійну граматику G' , причому кожному ліволінійному виведенню слова $p \in L(G)$ відповідає правостороннє виведення того ж слова p в граматиці G' і навпаки. Значить,

$$L(G) = L(G').$$

Теорема доведена.

Отже, надалі достатньо розглядати лише праволінійні граматики, оскільки їх основні властивості переносяться і на ліволінійні граматики.

Нехай $G = (V_T, V_H, \sigma, P)$ — деяка праволінійна граматика з термінальним алфавітом $V_T = \{a_1, a_2, \dots, a_n\}$. Вилучимо з множини P правила, що мають вигляд $\psi \rightarrow e$ і $\psi \rightarrow \tau$, де e — пусте слово, а $\tau, \psi \in V_H$ (в наступному параграфі буде докладно описана процедура такого вилучення). Одержана в результаті такого перетворення граматика G^* породжує з точністю до пустого слова e ту ж саму мову, що і граматика G , тобто $L(G^*) = L(G) \setminus \{e\}$. Поставимо тепер у відповідність кожному правилу $\psi_i \rightarrow u_i \tau_i$, де $u_i = a_{i1} a_{i2} \dots a_{ik} \in F(V_T)$, сукупність правил

$$P(i) = \begin{cases} \psi_i \rightarrow a_{i1} \psi_{i1}, \\ \psi_{i1} \rightarrow a_{i2} \psi_{i2}, \\ \dots \\ \psi_{ik-2} \rightarrow a_{ik-1} \psi_{ik-1}, \\ \psi_{ik-1} \rightarrow a_{ik} \tau_i. \end{cases} \quad (8.1.2)$$

шляхом введення множини додаткових нетермінальних символів $V(i) = \{\psi_{i1}, \psi_{i2}, \dots, \psi_{ik-1}\}$, $\psi_j \notin V_H$ ($j = 1, 2, \dots, k - 1$).

Множина правил, яка відповідає заключній підстановці $\psi_i \rightarrow \tau_i$, може бути одержана з (8.1.2), якщо замінити правило

$$\psi_{ik-1} \rightarrow a_{ik} \tau_i$$

правилом $\psi_{ik-1} \rightarrow a_{ik}$. При цьому вважається, що для різних правил системи P множини додаткових нетермінальних символів не перетинаються: $V(i) \cap V(j) = \emptyset$, $i \neq j$.

Очевидно, що граматика $G^a = (V_T, V_H, \sigma, P)$, де

$$V_H = V_H \cup (\bigcup_i V(i)), \quad P = \bigcup_i P(i),$$

еквівалентна граматиці G .

Означення 8.1.4. Граматика G^a , кожне правило якої має вигляд $\psi \rightarrow z$, де $z = a$ або $z = at$, $a \in V_T$, $t \in V_H$, називається **автоматною** або **граматикою зі скінченним числом станів**.

Граматика, яка розглядалась при доведенні теореми 8.1.2, не є автоматною граматикою, але цій граматиці еквівалентна граматика, що включає такі правила підстановки:

$$\begin{aligned}\sigma &\rightarrow a\psi, \\ \psi &\rightarrow a\psi, \\ \psi &\rightarrow ct, \\ t &\rightarrow bt, \\ t &\rightarrow b.\end{aligned}$$

З наведених вище міркувань випливає, що для довільної праволінійної граматики G знайдеться автоматна граматика G^a , яка породжує з точністю до пустого слова e ту ж саму мову, що і граматика G , тобто $L(G^a) = L(G) \setminus \{e\}$.

Існує тісний зв'язок між автоматними граматиками і скінченною автоматаами, чим і пояснюється термін “автоматна граматика”, або “граматика зі скінченним числом станів”. Дійсно, з кожною автоматною граматикою $G^a = (V_T, V_H, \sigma, P)$ асоціюється скінчений автомат без виходів $A = (A = V_H \cup \{\phi\} (\phi \notin V_H), X = V_T, \phi, \{a_0 = \sigma\}, F = \{\phi\})$, де функція переходів f задана графом переходів Γ . Граф Γ будується за такими правилами.

Кожному нетермінальному символу ψ із V_H граматики G^a відповідає вершина графа Γ , яка помічена символом ψ ; крім того, є ще одна вершина, помічена символом заключного стану — ϕ . Единому початковому стану σ автомата A відповідає вершина, помічена символом σ . Нарешті, для кожного незаключного правила $\psi \rightarrow at$ проводиться ребро з вершини ψ у вершину t , причому це ребро помічається термінальним символом $a \in V_T$; для кожного заключного правила підстановки $\psi \rightarrow a$ проводиться ребро, яке помічене символом a , із вершини ψ у вершину ϕ . Автомат функціонує звичайним чином, так як це було описано в розділі 6, але слід зауважити, що побудований за цими правилами автомат буде, взагалі кажучи, недетермінованим. За теоремою про детермінізацію для даного скінченного автомата можна побудувати еквівалентний йому детермінований скінчений автомат. Очевидно, що слово r належить мові $L(A)$, що представлена в автоматі A заключним станом ϕ , тоді і тільки тоді, коли слово r виводиться в граматиці G . Отже,

$$L(A) = L(G^a).$$

Враховуючи одержаний факт і за основною теоремою теорії скінчених автоматів, ми довели таку теорему.

Теорема 8.1.3. *Множина всіх мов, породжених автоматними граматиками, збігається з множиною всіх регулярних мов, які представляються в детермінованих скінчених автоматах.*

З цієї теореми випливає, що такі властивості, як еквівалентність лівлінійних і праволінійних, а також і автоматних граматик, алгоритмічно розв'язні, оскільки проблема зводиться до проблеми еквівалентності скінчених автоматів, яка є алгоритмічно розв'язною.

Задачі і вправи

1. Побудуйте в алфавіті $X = \{0, 1\}$ граматику, правильними словами якої є парні числа, якщо слово в алфавіті X розглядати як натуральне число в двійковій системі.

2. Наведіть приклад граматики, що породжує слова в алфавіті $X = \{a, b\}$, яка є:

- а) бс-граматикою;
- а) кв-граматикою;
- а) лінійною граматикою;
- а) праволінійною граматикою;
- а) лівлінійною граматикою.

3. Якими граматиками є наведені нижче граматики, що породжують слова в алфавіті $X = \{a, b, \dots, z\}$ за такими схемами:

а)

$$P = \begin{cases} \sigma \rightarrow a\sigma z, \\ \sigma \rightarrow b, \\ \sigma \rightarrow cb; \end{cases}$$

б)

$$P = \begin{cases} \sigma \rightarrow a\psi a, \\ \sigma \rightarrow b, \\ a\psi \rightarrow \psi a, \\ b\psi \rightarrow ab; \end{cases}$$

в)

$$P = \begin{cases} \sigma \rightarrow a\psi, \\ \sigma \rightarrow b, \\ \psi \rightarrow \psi a, \\ \psi \rightarrow ab ? \end{cases}$$

4. Побудуйте скінчений автомат, який представляє слова мови, що породжується граматикою “в” з попереднього прикладу.

5. Покажіть, що проблема еквівалентності двох автоматних граматик алгоритмічно розв'язна.

6. Побудуйте алгоритм, який за двома заданими автоматними граматиками — G і G' — буде граматику G'' що породжує слова мови

$$L(G) \cap L(G').$$

7. Побудуйте приклад мови, яка не може бути породжена жодною автомматичною граматикою.

§ 8.2. КОНТЕКСТНО-ВІЛЬНІ ГРАМАТИКИ ТА ЇХ ВЛАСТИВОСТІ

1. КОНТЕКСТНО-ВІЛЬНІ ГРАМАТИКИ І РІВНЯННЯ

З визначення кв-граматики (кв-мови) випливає, що всяка кв-граматика (кв-мова) є бс-граматикою (бс-мовою), але обернене твердження хибне. Відомі приклади бс-мов, для яких не існує кв-граматики, що породжує цю мову. Зокрема, такою мовою є мова з прикладу 8.1.1.

Нехай $G = (V_T, V_H, s, P)$ — довільна кв-граматика. *Лівим виведенням* слова $p \in L(G)$ в граматиці G називається виведення, в якому всяке правило граматики G , що застосовується на i -му кроці до слова $z_i = u\psi v_i$, застосовується до найлівішого нетермінального символу в слові z_i ($i = 1, 2, \dots, n - 1$).

Аналогічно визначається *праве виведення* слова $p \in L(G)$ в граматиці G , з тією різницею, що на i -му кроці виведення правила із P застосовуються до найправішого нетермінального символу в слові z_i ($i = 1, 2, \dots, n - 1$).

Очевидно, що кожному повному виведенню слова p в кв-граматиці відповідає єдине ліве (праве) виведення того ж слова p .

Приклад 8.2.1

Нехай $G = (\{a, b\}, \{\sigma, \tau\}, \sigma, P)$ — кв-граматика, де P складається з таких правил: $\{\sigma \rightarrow \sigma\tau, \tau \rightarrow \tau\tau, \sigma \rightarrow a, \tau \rightarrow b\}$.

Тоді виведення

$$(\sigma, \sigma\tau, a\tau, a\tau\tau, ab\tau\tau, ab\tau, abbb)$$

є лівим виведенням слова $p = abbb$ в граматиці G .

Виведення

$$(\sigma, \sigma\tau, \sigma\tau\tau, \sigma\tau\tau\tau, \sigma\tau\tau b, \sigma\tau b, \sigma bbb, abbb)$$

того ж слова $p = abbb$ в граматиці G є правим виведенням. \blacktriangleleft

Кожне повне виведення слова $p \in F(G)$ в кв-граматиці $G = (V_T, V_H, \sigma, P)$ в результаті деякої перестановки правил, що ви-

користовуються в цьому виведенні, може бути перетворене в ліве (праве) виведення. Ліве (праве) виведення приймають за *канонічну форму* виведення в кв-граматиці.

Кв-граматика $G = (V_T, V_H, \sigma, P)$ називається *визначеною*, якщо кожне правильне слово $p \in F(G)$ має єдине ліве (праве) виведення. В протилежному випадку граматика G називається *невизначеною*.

Поняття визначеності і невизначеності кв-граматики відіграють важливу роль у задачі автоматизації програмування. Якщо кв-граматика невизначена, тобто для деякого слова p із $L(G)$ існують різні ліві (праві) виведення, то виникає проблема: яке виведення слід розглядати, щоб правильно обробити дане слово p . Тому ці поняття належать до основних понять теорії кв-мов [13].

Узагальненою кв-граматикою (*укв-граматикою*) називається така граматика $G = (V_T, V_H, \sigma, P)$, кожне правило системи якої має вигляд $\psi \rightarrow z$, де $z \in F(V_T \cup V_H)$, $\psi \in V_H$.

Теорема 8.2.1. Для довільної укв-граматики $G = (V_T, V_H, \sigma, P)$ можна побудувати кв-граматику $G' = (V_T, V_H, \sigma, P')$ таку, що $L(G') = L(G) \setminus \{e\}$, причому якщо $e \notin L(G)$, то $L(G) = L(G')$ (тобто в цьому випадку $G \sim G'$).

Д о в е д е н н я. Нехай $G = (V_T, V_H, \sigma, P)$ — деяка укв-граматика. Побудуємо індуктивно множину $M \subseteq V_H$ таким чином, що $\psi \in M$, якщо в системі P є правило $\psi \rightarrow e$. Якщо $\psi_1, \psi_2, \dots, \psi_k \in M$ і в систему P входить правило підстановки $\psi \rightarrow \psi_1\psi_2\dots\psi_k$, то $\psi \in M$. Внаслідок скінченності алфавіту V_H і системи правил P множина M може бути побудована за скінченне число кроків. Правила, які використовувались при побудові множини M , назовемо *правилами, в яких зникає права частина*. Тепер кожному правилу $p: \psi \rightarrow z$ із P з непустою правою частиною z поставимо у відповідність систему $S(p)$ всіх правил $\psi \rightarrow z'$, де слово $z' \neq e$ і z' одержане із слова z в результаті вилучення деякої (можливо пустої) сукупності символів із M .

Розглянемо кв-граматику $G' = (V_T, V_H, \sigma, P')$, де $P' = \bigcup_p S(p)$, для всіх правил $p \in P$ з непустими правими частинами.

Покажемо, що мови $L(G)$ і $L(G')$ збігаються з точністю до пустого слова. Дійсно, застосування кожного правила $p' \in S(p)$, такого, що $p' \notin P$, можна замінити застосуванням правила p і деякої послідовності правил, в яких зникає права частина. Отже, має місце включення $L(G') \subseteq L(G)$.

Покажемо, що $L(G') \setminus \{e\} \subseteq L(G)$. Нехай x — довільне непусте слово із $L(G)$ і $W = \{z_0, z_1, \dots, z_n\}$ — ліве виведення слова $x = z_n$ в укв-граматиці G , а T — дерево, яке відповідає виведенню W .

Припустимо, що на деяких кроках виведення W застосовуються правила, що мають вигляд

$$\psi \rightarrow e. \quad (8.2.1)$$

В протилежному випадку W є повним виведенням в граматиці G' і $x \in L(G')$. Послідовно вилучаючи застосування правил, що мають вигляд (8.2.1), можна перетворити виведення W в виведення W' того ж слова x в граматиці G' .

Дійсно, нехай у виведенні W правило типу (8.2.1) застосовується на i -му кроці ($1 \leq i \leq n$) до входження (ψ, q) символу ψ в слово $z_{i-1} = u\psi u_1$, де $l(u) = q - 1$. Очевидно, що $i \neq 1$, оскільки x — непусте слово. З цих же причин хоча б на одному із кроків, які передують i -му кроці, застосовується правило

$$\psi \rightarrow z, \quad (8.2.1')$$

таке, що $l(z) > 1$, у протилежному ж випадку $z_i = x = e$. Тоді можна вказати найбільший номер $m < i$, такий, що на m -му кроці виведення W застосовується правило p_m типу (8.2.1'): $p_m: \tau \rightarrow u\psi v$. При цьому слово $z_m = u' u \psi' v \in W'$ і в дереві T існує шлях із вершини s , позначеної (ψ', q') , у вершину (ψ, q) , де (ψ', q') — входження виділеного символу ψ' в слово z_m , (ψ, q) — входження ψ в слово z_i . Нехай

$$\psi' = \psi_0, \psi_1, \dots, \psi_r = \psi \quad (8.2.1'')$$

є послідовність нетермінальних символів, що позначають вершини шляху s . Із максимальності номера m випливає, що з кожної вершини ψ_j ($j = 0, 1, \dots, r - 1$) виходить точно одна дуга, і кожному такому переходу відповідає застосування правила

$$\Psi_j \rightarrow \Psi_{j+1} \quad (8.2.1a)$$

на m_j -му кроці виведення W , $m < m_j < i$.

Враховуючи застосування правила $\psi \rightarrow e$ на i -му кроці виведення W , маємо, що правила (8.2.1a) належать до числа правил, в яких зникає права частина, і всі символи, які входять в послідовність (8.2.1''), належать множині M . Це означає, зокрема, що сукупність правил $S(p_m) \subseteq P'$ включає правило $p'_m: \tau \rightarrow uv$.

Перебудуємо тепер виведення W таким чином: на m -му кроці замість правила p_m застосуємо правило p'_m і вилучимо наступні m_j -і кроки виведення ($j = 0, 1, \dots, r$), пов'язані з перетворенням символу ψ' , що зник, а також i -й крок виведення W . Очевидно, що така перебудова не впливає на виведення скінченного слова $x \in L(G)$, в той час як число застосувань правил (8.2.1) зменшиться на одиницю порівняно з виведенням W . Продовжуючи описаний процес, перетворимо виведення W слова x в уявні граматиці G у виведення W' того ж слова, на кожному кроці

якого застосовуються правила із P' , так що застосування правил, що мають вигляд (8.2.1) повністю виключається.

Таким чином, виведення W' слова $x \in L(G')$. Внаслідок довільності непустого слова x із $L(G)$ випливає справедливість включення $(L(G) \setminus \{e\}) \subset L(G')$.

Теорема доведена.

З доведеної теореми випливає, що всяку укв-граматику $G = (V_T, V_H, \sigma, P)$ можна звести до еквівалентної їй кв-граматики $\underline{G} = (V_T, V_H, \underline{\sigma}, \underline{P})$, такої, що для будь-якого нетермінального символу $\psi \in V_H \setminus \{\sigma\}$ схема \underline{P} не містить правил $\psi \rightarrow e$ і аксіома $\underline{\sigma}$ не входить у праві частини правил підстановки.

Дійсно, для граматики G існує кв-граматика $G' = (V_T, V_H', \sigma, P)$, що задовольняє умови теореми 8.2.1. Граматика \underline{G} будується за граматикою G' таким чином. Вибирається як аксіома новий нетермінальний символ $\underline{\sigma} \notin V_H$ так, що $V_H = V_H' \cup \{\underline{\sigma}\}$. Для системи правил P виконується включення $P' \subset P$. Крім того, для кожного правила $\sigma \rightarrow z$ із P' включається правило $\underline{\sigma} \rightarrow z$. Система P включає правило $\underline{\sigma} \rightarrow e$, якщо $e \in L(G)$. Зазначимо, що аксіома $\underline{\sigma}$ не входить у праві частини правил системи P . Звідси, оскільки в P для довільного $\psi \neq \underline{\sigma}$ відсутні правила $\psi \rightarrow e$, слова $z \neq e$, які породжуються при будь-якому виведенні в граматиці G , не вкорочуються. Отже, граматика \underline{G} називається *формою граматики G, що не укорочує слова*.

Приклад 8.2.2

Нехай $G = (\{a, b\}, \{\sigma, \psi_1, \psi_2, \psi_3\}, \sigma, P)$, де

$$P_1 = \begin{cases} \sigma \rightarrow a\sigma b, \\ \sigma \rightarrow e, \\ \psi_1 \rightarrow \psi_2\psi_3, \\ \psi_2 \rightarrow \sigma\psi_3, \\ \psi_3 \rightarrow e. \end{cases}$$

Формою укв-граматики G , що не укорочує слова, є укв-граматика $\underline{G}_1 = (\{a, b\}, \{\tau, \sigma, \psi_1, \psi_2, \psi_3\}, \tau, P_1)$, де

$$\underline{P}_1 = \begin{cases} 1: \tau \rightarrow e, & 7: \psi_1 \rightarrow \psi_2, \\ 2: \sigma \rightarrow a\sigma b, & 8: \psi_1 \rightarrow \psi_3, \\ 3: \tau \rightarrow a\sigma b, & 9: \psi_2 \rightarrow \sigma\psi_3, \\ 4: \sigma \rightarrow ab, & 10: \psi_2 \rightarrow \psi_3, \\ 5: \tau \rightarrow ab, & 11: \psi_2 \rightarrow \sigma. \\ 6: \psi_1 \rightarrow \psi_2\psi_3, & \end{cases}$$

Наслідок 8.2.1. Всяку укв-граматику G можна перетворити в еквівалентну їй укв-граматику \tilde{G} , яка є формою граматики G , що не вкорочує слова.

З теореми 8.2.1 випливає також, що відмова від вимоги непустоти правих частин правил підстановки кв-граматики, на відміну від бс-граматик, не приводить до суттєвого розширення класу мов, які ними породжуються. Тому поняття укв- і кв-граматик часто не розрізняються і під кв-граматикою слід розуміти довільну укв-граматику, задану у формі, що не вкорочує слова.

З множини правил підстановки P граматики $G = (V_T, V_H, \Sigma, P)$ можна вилучити всі правила, що мають вигляд

$$\psi \rightarrow \tau, \quad (8.2.2)$$

де $\tau, \psi \in V_H$. Для цього достатньо побудувати всі можливі виведення

$$\psi \xrightarrow{G}^* \tau. \quad (8.2.2')$$

Внаслідок скінченності множин V_H і P , а також невкорочуваності слів при виведенні в граматиці G , побудова всіх виведень типу (8.2.2') виконується за скінченне число кроків. Після цього для кожного із виведень $\psi: \xrightarrow{G}^* \tau$ і довільного правила $\tau \rightarrow z$ із P , де $z \notin V_H$, приєднується нове правило $\psi \rightarrow z$, після чого всі правила, що мають вигляд (8.2.2'), можуть бути вилучені. В результаті приходимо до граматики \tilde{G} , яка еквівалентна граматиці G і не має правил, що мають вигляд (8.2.2).

Приклад 8.2.3

Розглянемо граматику G_1 із попереднього прикладу. За схемою правил P_1 побудуємо всі виведення типу (8.2.2) в граматиці G_1 :

$$\psi_1 \rightarrow^* \psi_2, \psi_1 \rightarrow^* \psi_3, \psi_1 \rightarrow^* \sigma, \psi_2 \rightarrow^* \psi_3, \psi_3 \rightarrow^* \sigma.$$

Для виведення $\psi_1 \rightarrow^* \psi_2$ і правила 9: $\psi_2 \rightarrow \sigma\psi_3$ будуємо множину $S_0 = \{\psi_1 \rightarrow \sigma\psi_3\}$, для виведення $\psi_1 \rightarrow^* \sigma$ і правил 2 і 4 — множину $S_1 = \{\psi_1 \rightarrow a\sigma b, \psi_1 \rightarrow ab\}$ і, нарешті, для виведення $\psi_2 \rightarrow^* \sigma$ і правил 2 і 4 — множину $S_2 = \{\psi_2 \rightarrow a\sigma b, \psi_2 \rightarrow ab\}$.

Тепер, об'єднавши множину P_1 з множиною $S_0 \cup S_1 \cup S_2$ і вилучивши правила 7, 8, 10 і 11, будуємо множину правил \tilde{P}_1 граматики $\tilde{G}_1 = (\{a, b\}, \{\tau, \sigma, \psi_1, \psi_2, \psi_3\}, \tau, \tilde{P}_1)$. ▲

З наведених визначень і теорем випливає, що для всякої кв-граматики існує приведена кв-граматика, яка задовільняє такі умови:

- 1) приведена граматика є граматикою, що не вкорочує слова;
- 2) аксіома не входить в праві частини правил підстановки;
- 3) в множину правил граматики не входять правила типу (8.2.2).

Отже має місце така теорема.

Теорема 8.2.2. Для довільної кв-граматики $G = (V_T, V_H, \sigma, P)$ існує еквівалентна їй приведена граматика $\tilde{G} = (V_T, \tilde{V}_H, \sigma, \tilde{P})$, де $\tilde{V}_H = V_H$ — алфавіт нетермінальних символів, одержаний із V_H шляхом вилучення всіх фіктивних символів граматики G , $\tilde{P} \subseteq P$ — система правил підстановки граматики G , що не включає фіктивних символів.

Ця теорема дає можливість без обмеження загальності розглядати лише приведені кв-граматики.

2. РІВНЯННЯ, КВ-МОВИ І МАГАЗИННІ АВТОМАТИ

За кв-граматикою $G = (V_T, V_H, \sigma, P)$, яка породжує слова в алфавіті $V = V_T \cup V_H$, можна побудувати систему рівнянь, що визначає ту ж саму мову $L(G)$ [61].

Дійсно, нехай $V_H = \{\psi_1, \psi_2, \dots, \psi_n\}$, де $\psi_i = \sigma$ — аксіома граматики G . Кожному нетермінальному символу $\psi_i \in V_H$ поставимо у відповідність рівняння

$$\psi_i ::= f_i(\psi_1, \psi_2, \dots, \psi_n),$$

де $f_i(\psi_1, \psi_2, \dots, \psi_n) = z_{i1} \cup z_{i2} \cup \dots \cup z_{ik}$ — вираз, який складається із всіх правих частин правил підстановки, що мають вигляд

$$\psi_i \rightarrow z_{ij} \in P \quad (j = 1, 2, \dots, k_i).$$

Таким чином, граматиці G буде відповідати система рівнянь

$$U_G = \{\psi_i ::= f_i(\psi_1, \psi_2, \dots, \psi_n) \mid i = 1, 2, \dots, n\}. \quad (8.2.3)$$

Розглянемо процес породження мови $L(G)$ за системою рівнянь (8.2.3), яка відповідає граматиці G . Нехай $Q = (\emptyset, \emptyset, \dots, \emptyset)$ — нульовий вектор значень змінних $\psi_1, \psi_2, \dots, \psi_n$, в якому значення змінних $\psi_i = \emptyset$. На такому нульовому наборі функції $f_i(\psi_1, \psi_2, \dots, \psi_n)$, які є правими частинами системи рівнянь (8.2.3), набувають значення $\alpha_i = f_i(\emptyset, \emptyset, \dots, \emptyset)$ для кожного $i = 1, 2, \dots, n$. Позначимо сукупність цих значень $\alpha_0 = (\alpha_{01}, \alpha_{02}, \dots, \alpha_{0n})$.

Тепер розглянемо значення $\alpha_{11} = f_1(\alpha_0)$, ..., $\alpha_{1n} = f_n(\alpha_0)$ функцій $f_i(\psi_1, \psi_2, \dots, \psi_n)$ на наборі α_0 , так що $\alpha_1 = (\alpha_{11}, \dots, \alpha_{1n})$. Продовжуючи рекурсивно цей процес далі, одержимо $\alpha_k = (\alpha_{k1}, \alpha_{k2}, \dots, \dots, \alpha_{kn})$, де $\alpha_{ki} = f_i(\alpha_{k-1}) = f_i(\alpha_{k-11}, \alpha_{k-12}, \dots, \alpha_{k-1n})$ для довільного $k = 1, 2, \dots$. Позначимо α_i^∞ множину $\bigcup_{k=1}^{\infty} \alpha_{ki}$ для довільного $i = 1, 2, \dots, n$; тоді набір $\alpha^\infty = (\alpha_1^\infty, \dots, \alpha_n^\infty)$ називається розв'язком системи рівнянь (8.2.3), а кожен компонент α_i^∞ — мовою, яка відповідає змінній ψ_i (або мовою типу АЛГОЛ [13]).

Мова α_1^∞ , яка відповідає аксіомі $\psi_1 = \sigma$, називається *мовою, що породжена системою рівнянь* (8.2.3). Неважко показати, що мова α_1^∞ , породжена системою (8.2.3), збігається з мовою $L(G)$, породженою граматикою G , тобто $\alpha_1^\infty = L(G)$ [61].

Крім того, $\alpha^\infty = (\alpha_1^\infty, \dots, \alpha_n^\infty)$ є мінімальним розв'язком системи (8.2.3), так що коли $L = (L_1, L_2, \dots, L_n)$ — довільний розв'язок цієї ж системи, то $\alpha_i^\infty \subseteq L_i$ для довільного $i = 1, 2, \dots, n$. Дійсно, $\alpha_{0i} \subset L_i$ ($i = 1, 2, \dots, n$). Користуючись методом індукції до описаного вище процесу, одержимо $\alpha_i^\infty \subset L_i$.

Теорема 8.2.3. *Мова L тоді і тільки тоді є кв-мовою, коли існує система рівнянь типу (8.2.3), яка має мінімальний розв'язок $\tilde{\alpha}^\infty = (\alpha_1^\infty, \alpha_2^\infty, \dots, \alpha_n^\infty)$, такий, що $\alpha_1^\infty = L$.*

Приклад 8.2.4

Розглянемо граматику $G = (\{a, b\}, \{\sigma\}, \sigma, P)$, де P має такі правила: $(\sigma \rightarrow a\sigma a, \sigma \rightarrow b)$. Цій граматиці відповідає система U_G , яка складається з одного рівняння

$$U_G = \{ \sigma ::= a\sigma a \cup b \}.$$

Побудуємо мову, яка породжується цією системою. Маємо
 $\alpha_0 = f(\emptyset) = a\emptyset a \cup b = \emptyset \cup b = b$;
 $\alpha_1 = f(\alpha_0) = a(\alpha_0)a \cup b = aba \cup b$;
 $\alpha_2 = f(\alpha_1) = a(\alpha_1)a \cup b = a(aba \cup b)a \cup b = a^2ba^2 \cup aba \cup b$;
 $\alpha_3 = f(\alpha_2) = a(\alpha_2)a \cup b = a^3ba^3 \cup a^2ba^2 \cup aba \cup b$
і т. д. Отже розв'язком системи є мова $L = \{a^nba^n \mid n = 0, 1, \dots\}$. \blacktriangleleft

У попередньому розділі при вивчені магазинних автоматів ми бачили, як будувати за кв-граматикою магазинний автомат, в якому представлена мова, породжена цією кв-граматикою, і як

будувати за заданим магазинним автоматом систему рівнянь, розв'язком якої є мова, що представлена в ньому. Отже, з алгоритмів аналізу і синтезу магазинних автоматів, а також доведеної теореми випливає такий наслідок

Наслідок 8.2.2. Мова L є кв-мовою тоді і тільки тоді, коли вона може бути представлена в деякому магазинному автоматі.

Задачі і вправи

1. Користуючись методами, описаними при доведенні теорем 8.2.1 і 8.2.2, побудуйте приведені форми граматик, що не вкорочують слова, для таких граматик:

а) $G = (\{a, b\}, \{\sigma, \psi_1, \psi_2, \psi_3\}, \sigma, P)$, де

$$P = \begin{cases} \sigma \rightarrow a\sigma b, \\ \sigma \rightarrow e, \\ \psi_1 \rightarrow \psi_2 \psi_3, \\ \psi_2 \rightarrow \sigma \psi_3, \\ \psi_3 \rightarrow e; \end{cases}$$

б) $G = (\{a, b, \dots, z\}, \{\sigma\}, \sigma, P)$, де

$$P = \begin{cases} \sigma \rightarrow a\sigma z, \\ \sigma \rightarrow b, \\ \sigma \rightarrow cb; \end{cases}$$

в) $G = (\{a, b\}, \{\sigma, \psi\}, \sigma, P)$, де

$$P = \begin{cases} \sigma \rightarrow a\psi a, \\ \sigma \rightarrow b, \\ a\psi \rightarrow \psi a, \\ b\psi \rightarrow ab; \end{cases}$$

г) $G = (\{a, b\}, \{\sigma, \psi\}, \sigma, P)$, де

$$P = \begin{cases} \sigma \rightarrow a\psi, \\ \sigma \rightarrow b, \\ \psi \rightarrow \psi a, \\ \psi \rightarrow ab. \end{cases}$$

2. Застосувавши алгоритм синтезу до граматик, побудованих у вправі 1, знайдіть відповідні магазинні автомати, які представляють мови, породжені цими граматиками.

ЧАСТИНА III

ЗАСТОСУВАННЯ

АЛГЕБРИ В КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЯХ

У даному розділі розглядаються деякі застосування основних понять і методів загальної алгебри, які використовуються в сучасних інформаційних технологіях. До таких технологій належать бази даних, бази знань, системи комп'ютерної геометрії, графічні системи і т. п.

§ 9.1. ПЕРЕТВОРЕННЯ У ВЕКТОРНИХ ПРОСТОРАХ

У даному параграфі розглядається застосування методів теорії груп для обґрунтування так званих *геометричних перетворень* лінійних векторних просторів. Ці перетворення — одні з основних у графічних комп'ютерних системах і входять як складові практично до будь-якої такої системи.

Нижче розглядатимуться здебільшого двовимірний простір L_2 і тривимірний простір L_3 , які ми відповідно будемо називати *площиною* і *простором*.

1. ПЕРЕМІЩЕННЯ ПЛОЩИНИ

Спочатку дамо деякі необхідні означення з теорії лінійних векторних просторів.

Відображення (оператор) $T: L_n \rightarrow L_n$ називається *лінійним*, якщо для довільних векторів u, v із L_n і довільних чисел a, b із основного поля P лінійного векторного простору L_n справедлива рівність $T(a \cdot u + b \cdot v) = a \cdot T(u) + b \cdot T(v)$.

З лінійної алгебри відомо, що кожному лінійному оператору T в L_n відповідає матриця, що називається *матрицею оператора* T , причому ця відповідність взаємно однозначна.

Лінійний оператор T в евклідовому лінійному просторі L_n називається **ортогональним**, якщо для будь-яких векторів u, v із L_n справедлива рівність $(T(u), T(v)) = (u, v)$, де (u, v) — скалярний добуток векторів u і v .

З розділу 2 ми знаємо, що будь-яка система з n лінійно незалежних векторів в n -вимірному векторному просторі становить базис цього простору. Якщо $\{v_1, v_2, \dots, v_n\}$ — базис n -вимірного векторного простору L_n над деяким полем P , то будь-який вектор v із L_n єдиним способом можна записати у вигляді суми:

$$v = \sum_{i=1}^n a_i \cdot v_i,$$

де $a_i \in P$. При цьому n -ка чисел (a_1, \dots, a_n) ототожнюється з вектором v і називається координатами даного вектора v в базисі $\{v_1, v_2, \dots, v_n\}$. Коли базисні вектори v_i , $i = 1, 2, \dots, n$, такі, що довжина $|v_i|$ вектора v_i дорівнює 1, то базис називається **ортонормованим**, а відповідні координати вектора v в цьому базисі — **декартовими координатами вектора v** .

В просторах L_2 і L_3 кожний вектор також розкладається за базисними векторами e_1, e_2, e_3 і задається у вигляді $u = (x, y, z)$, де x, y, z — декартові координати. Цей запис означає, що $u = x \cdot e_1 + y \cdot e_2 + z \cdot e_3$. Аналогічно задаються і координати точки A , які ототожнюються з координатами вектора r , проведеного з початку координат 0 в точку A . Вектор r називається **радіус-вектором** точки A . Той факт, що x, y, z — координати точки A , записується так: $A(x, y, z)$.

Переміщенням простору L_n називається таке відображення

$$F: L_n \rightarrow L_n,$$

яке зберігає відстані між точками, тобто якщо F — переміщення і A, B — довільні точки із L_n , то $|AB| = |A'B'|$, де $A' = F(A)$, $B' = F(B)$, а $|AB|$ — відстань між точками A і B .

Приклади переміщень площини добре відомі:

- паралельний перенос на вектор r — T_r ;
- поворот навколо точки O на кут θ — TR_θ ;
- симетрія відносно прямої l — S_l ;
- ковзна симетрія — S'_l .

Розглянемо кожне з цих перетворень окремо.

Відображення $T_r: L_n \rightarrow L_n$ називається **паралельним переносом**, якщо $T_r(v) = v + r$, де r — деякий фіксований вектор, а v — довільний вектор із L_n . В координатній формі це перетворення записується так:

$$T_r(v) = v' = (x + x_0, y + y_0, \dots, z + z_0),$$

де $v = (x, y, \dots, z)$, $r = (x_0, y_0, \dots, z_0)$.

Безпосередньо з означення T_r випливають такі його властивості.

B1. Перетворення T_r не є лінійним.

Дійсно,

$$\begin{aligned} T_r(a \cdot u + b \cdot v) &= a \cdot u + b \cdot v + r, \\ a \cdot T_r(u) + b \cdot T_r(v) &= a \cdot (u + r) + b \cdot (v + r) = \\ &= a \cdot u + b \cdot v + (a + b) \cdot r. \end{aligned}$$

B2. $T_r * T_r = T_{r+r}$. Дійсно,

$$T_r * T_r(v) = T_r(T_r(v)) = T_r(v + r) = v + r + r' = T_{r+r'}(v)$$

для довільного v із L_n .

Теорема 9.1.1. *Множина всіх паралельних переносів є абелевою групою відносно операції добутку відображення.*

Д о в е д е н н я. Із В2 випливає замкнутість операції добутку, а із В2 і аксіом лінійного векторного простору L_n — асоціативність і комутативність цієї операції, а також існування переносів $T_r(-v)$ і T_0 , які є оберненим елементом до $T_r(v)$ і нулем групи відповідно.

Групу всіх паралельних переносів L_n позначають $T(n)$.

Поворотом площини відносно деякої точки O на кут θ називається відображення $TR_\theta: L_2 \rightarrow L_2$, що переводить точку $A(x, y)$ в точку $A'(x', y')$ шляхом повороту L_2 відносно точки O на кут θ . З

лінійної алгебри відомо, що перетворення повороту є лінійним перетворенням. Знайдемо його матрицю.

Нехай точка O збігається з початком координат. Тоді, як видно з рис. 9.1.1, можна записати:

$$\begin{aligned} x' &= |OA| \cdot \cos(\theta + \theta'), \\ y' &= |OA| \cdot \sin(\theta + \theta'). \end{aligned}$$

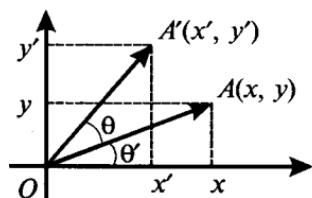


Рис. 9.1.1. Поворот площини

Виконуючи тоді ж перетворення над одержаними виразами, маємо:

$$\begin{aligned} x' &= |OA| \cdot (\cos \theta \cdot \cos \theta' - \sin \theta \cdot \sin \theta') = |OA| \cdot \cos \theta \cdot \cos \theta' - \\ &- |OA| \cdot \sin \theta \cdot \sin \theta' = x \cdot \cos \theta - y \cdot \sin \theta. \end{aligned}$$

Аналогічно одержуємо $y' = x \cdot \sin \theta + y \cdot \cos \theta$. Отже, матриця перетворення повороту в L_2 має такий вигляд:

$$TR_\theta = \begin{vmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{vmatrix}.$$

Користуючись матрицею повороту, неважко встановити такі властивості перетворення повороту.

ВП1. Перетворення повороту ортогональне.

Дійсно, якщо $u = (x, y)$, $v = (x', y')$ і поворот виконується на кут θ , то

$$\begin{aligned}TR_{\theta}(u) &= (x \cdot \cos \theta - y \cdot \sin \theta, x \cdot \sin \theta + y \cdot \cos \theta), \\TR_{\theta}(v) &= (x' \cdot \cos \theta - y' \cdot \sin \theta, x' \cdot \sin \theta + y' \cdot \cos \theta).\end{aligned}$$

Тоді

$$\begin{aligned}(TR_{\theta}(u), TR_{\theta}(v)) &= (x \cdot x' \cdot \cos^2 \theta - x' \cdot y \cdot \cos \theta \cdot \sin \theta - \\&- x \cdot y' \cdot \cos \theta \cdot \sin \theta + y \cdot y' \cdot \sin^2 \theta + x \cdot x' \cdot \sin^2 \theta + x \cdot y' \cdot \cos \theta \cdot \\&\cdot \sin \theta + x' \cdot y \cdot \cos \theta \cdot \sin \theta + y \cdot y' \cdot \cos^2 \theta = x \cdot x'(\cos^2 \theta + \sin^2 \theta) + \\&+ y \cdot y'(\cos^2 \theta + \sin^2 \theta) = x \cdot x' + y \cdot y' = (u, v).\end{aligned}$$

Властивість доведена.

З цієї властивості і того, що визначник матриці повороту дорівнює одиниці, випливає: існує обернене перетворення TR_{θ}^{-1} і матриця цього перетворення може бути одержана з матриці TR_{θ} шляхом її трансформування, тобто

$$TR_{\theta}^{-1} = \begin{vmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{vmatrix} = \begin{vmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{vmatrix}$$

$$\text{i } TR_{\theta}^{-1} = TR_{(-\theta)}.$$

ВП2. $TR_{\theta} * TR_{\theta'} = TR_{\theta+\theta'}$.

Пропонується довести читачеві як просту вправу.

Очевидно, що $T_r * TR_{\theta} \neq TR_{\theta} * T_r$.

Теорема 9.1.2. *Множина всіх поворотів площини відносно деякої фіксованої точки O в цій площині є абелевою групою.*

Доведення пропонується як проста вправа (див. вправу 4 в кінці розділу).

Пропонується також показати, що множина всіх поворотів не буде абелевою групою.

Точки $A(x, y)$ і $B(x', y')$ площини L_2 називаються *симетричними* відносно деякої прямої l цієї площини, якщо відрізок AB перпендикулярний до прямої l і ділиться цією прямою в точці перетину пополам. Перетворенням симетрії S_l називається відображення $S_l : L_2 \rightarrow L_2$, що переводить кожну точку $A(x, y)$ в симетричну їй точку $B(x', y')$ відносно прямої l . Пряма l називається *віссю симетрії*.

Знайдемо, чому дорівнюють координати точки $B(x', y')$, якщо відомі координати точки $A(x_0, y_0)$. Розглянемо спочатку випадок, коли вісь симетрії проходить через початок координат, тобто має вигляд $y = ax$. Нехай пряма l' , перпендикулярна до l , теж проходить через початок координат, тобто $y = (-1/a)x$. Тоді, якщо $A(x_0, y_0)$ лежить на прямій l' , то симетрична їй точка $B(x', y')$,

очевидно, має координати $x' = -x_0$, $y' = -y_0$. Якщо ж точка $A(x_0, y_0)$ лежить на прямій $y = (-1/a)x + b$, то знайдемо координати точки перетину прямих l і l' . Вони знаходяться з рівняння

$$ax' = (-1/a)x' + b$$

і, значить,

$$x' = ba/(a^2 + 1), \quad y' = ba^2/(a^2 + 1),$$

а B має координати

$$x' = -x_0 + ba/(a^2 + 1), \quad y' = -y_0 + ba^2/(a^2 + 1). \quad (9.1.1)$$

Нарешті розглянемо випадок, коли l має загальний вигляд: $y = ax + c$. Нехай $A(x_0, y_0)$, тоді перенесемо пряму l паралельно самій собі в початок координат. Нові координати точки A будуть мати вигляд

$$\begin{aligned} x_1 &= -x_0 + ca/(a^2 + 1), \\ y_1 &= -y_0 + ca^2/(a^2 + 1). \end{aligned}$$

Тепер залишається одержані координати підставити в (9.1.1) замість x_0 і y_0 . Виконуючи цю підстановку, одержуємо остаточні формули:

$$\begin{aligned} x' &= x_0 + (ba - ca)/(a^2 + 1), \\ y' &= y_0 + (ba^2 - ca^2)/(a^2 + 1). \end{aligned} \quad (9.1.2)$$

З означення симетрії і формул (9.1.2) випливають такі властивості симетрії.

ВС1. Перетворення симетрії нелінійне.

ВС2. $S_l * S_l = \varepsilon$, де ε — тотожне перетворення.

Ковзною симетрією S'_l називається добуток перетворень $T_r * S_l$, тобто

$$S'_l : L_2 \rightarrow L_2,$$

де $S'_l(v) = T_r * S_l(v) = S_l(T_r(v)) = S_l(v + r)$ і T_r — паралельний перенос на вектор r , паралельний осі l . У цьому випадку

$$T_r * S_l = S_l * T_r.$$

Дійсно,

$$\begin{aligned} T_r * S_l(v) &= S_l(v + r) = S_l(x + x_0, y + y_0) = \\ &= (x + x_0 + (ba - ca)/(a^2 + 1), \quad y + y_0 + (ba^2 - ca^2)/(a^2 + 1)); \\ S_l * T_r(v) &= T_r(S_l(v)) = T_r(x + (ba - ca)/(a^2 + 1), \\ &\quad y + (ba^2 - ca^2)/(a^2 + 1)) = (x + x_0 + (ba - ca)/(a^2 + 1), \\ &\quad y + y_0 + (ba^2 - ca^2)/(a^2 + 1)). \end{aligned}$$

Повну характеристику переміщень площини дає теорема Шаля.

Теорема Шаля. *Всяке переміщення в площині — це одне з трьох перетворень:*

- 1) паралельний перенос;
- 2) поворот;
- 3) ковзна симетрія (зокрема, при $r = 0$ і симетрія).

Доведення цієї теореми опускається, але при необхідності його можна знайти в [2].

2. ГРУПИ ПЕРЕТВОРЕНЬ ПЛОЩИНІ

Переміщення площини розбиваються на дві групи — *переміщення першого роду*, до яких належать перенос і поворот, і *переміщення другого роду*, до яких належать симетрія і ковзна симетрія. Обґрунтування такого розділення дається твердженням.

Твердження 9.1.1. *Добуток двох переміщень першого роду є переміщенням першого роду, добуток переміщень першого і другого роду є переміщенням другого роду і добуток переміщень другого роду є переміщенням першого роду.*

Доведення пропонується як проста вправа (див. задачу 3 в кінці розділу).

Теорема 9.1.3. *Множина всіх переміщень площини є групою відносно операції добутку відображень.*

Доведення. Очевидно, що коли перетворення F і G зберігають відстані між точками, то перетворення $F * G$ також буде зберігати відстані між точками. Асоціативність випливає з асоціативності операції добутку відображень (див. § 1.1, 4.9), а totожне перетворення, очевидно, є переміщенням. Нарешті, як випливає з ВП1, ВС2 і теореми 9.1.1, перетворення, обернене до перетворення переміщення, теж буде переміщенням.

Теорема доведена.

Групу всіх переміщень L_2 позначають $E(2)$. Група $E(2)$ має нескінченне число підгруп. Дійсно, згідно з твердженням 9.1.1 підгрупою будуть всі переміщення першого роду. Цю підгрупу позначають $E_0(2)$. Якщо F — переміщення першого роду, а G — переміщення другого роду, то згідно з твердженням 9.1.1 переміщення $G * F * G^{-1}$ є переміщенням першого роду, тобто підгрупа $E_0(2)$ є нормальним дільником або інваріантною підгрупою групи $E(2)$. Неважко помітити, що існує лише два класи по цій підгрупі: вона сама і клас переміщення другого роду. Отже, $E_0(2)$ має індекс 2 в групі $E(2)$ і фактор-група $E(2)/E_0(2)$ є скінченною циклічною групою другого порядку.

Розглянемо тепер групу $E_0(2)$. Ця група має нескінченне число підгруп поворотів: сукупність всіх поворотів відносно якої-небудь фіксованої точки площини становить групу. Всі ці групи комутативні.

Сукупність всіх поворотів підгрупою не буде. Дійсно, поворот навколо точки O на кут α , а потім поворот навколо точки O' на кут $2\pi - \alpha$ буде паралельним переносом (див. вправу 8 в кінці розділу).

Теорема 9.1.4. *Множина всіх паралельних переносів буде підгрупою $E_0(2)$.*

Оскільки будь-який перенос однозначно характеризується довжиною і напрямком вектора переносу, то група всіх паралельних переносів відносно прямої l (це переноси, вектори яких паралельні прямій l) буде ізоморфною групі всіх дійсних чисел відносно звичайної операції додавання. Цю групу позначають $T(2)$.

Теорема 9.1.5. *Група $T(2)$ — нормальнй дльник групи $E_0(2)$.*

Доведення обох теорем пропонуються як прості вправи.

3. ПЕРЕМІЩЕННЯ ПРОСТОРУ

Аналогічно переміщенню площини дається означення переміщення простору. Це — відображення $T: L_3 \rightarrow L_3$, що зберігає відстані між точками. Переміщення простору теж добре відомі:

- паралельний перенос на вектор $r = T_r$;
- поворот навколо осі l на кут $\theta = TR_\theta^l$;
- гвинтове переміщення — $S'_{l,\theta}$;
- симетрія відносно площини $P = S_P$;
- ковзна симетрія;
- поворотна симетрія.

Паралельний перенос був визначений раніше.

Поворот TR_θ^l навколо осі l на кут θ — це перетворення, що являє собою поворот кожної точки простору в площині, яка проходить через цю точку і перпендикулярна до прямої l , на кут θ навколо точки перетину l з площину.

Вісь l і кут повороту θ задають поворот неоднозначно, тому що один і той же результат вийде, коли виконати поворот навколо l на кут θ в одному напрямку і на кут $2\pi - \theta$ — в другому. Для забезпечення однозначності на площині вибирають додатний напрямок повороту — поворот проти годинникової стрілки, і від’ємний — за годинниковою стрілкою. В просторі це виконується так: вибирають на осі повороту деякий напрямок і вважають, що поворот додатний відносно даного напрямку на осі, коли довільна точка повертається в своїй площині проти годин-

никової стрілки для спостерігача, який стоїть уздовж осі так, що напрямок від його ніг до його голови збігається з напрямком, який був вибраний на осі.

Поворот TR_n' називається **перекиданням**. Очевидно, що $TR_n' * TR_n' = \epsilon$, де ϵ — тотожне перетворення.

Нехай вісь повороту l задана деяким одиничним напрямним вектором $n = (n_1, n_2, n_3)$ і $0 \leq \theta \leq 2\pi$. Знайдемо матрицю перетворення TR_θ' в L_3 . Нехай $TR_\theta'(v) = v'$ відносно осі n і v — вектор, кінець якого знаходиться в точці P , v' — вектор, кінець якого знаходиться в точці P' (рис. 9.1.2).

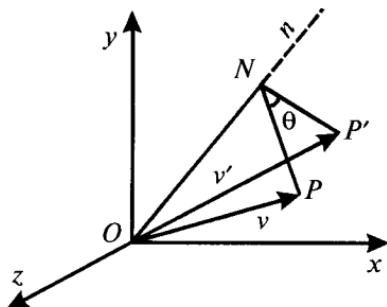


Рис. 9.1.2

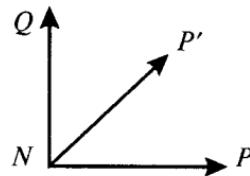


Рис. 9.1.3

Для того щоб знайти матрицю TR_θ' , виразимо v' через v , n , q і θ . З аналітичної геометрії відомо, що коли вектор v_N , кінець якого знаходиться в точці N , лежить на осі n , то $v_N = (n, v)n$, де (n, v) — скалярний добуток векторів n і v . Нехай v_{NP} — вектор, який з'єднує точки N і P . Тоді

$$v_{NP} = v - v_N = v - (v, n)n = [[n * v] * n],$$

де $[n * v]$ — векторний добуток. Розглянемо, який вигляд має поворот, коли дивитися з точки N в напрямку початку координат O (див. рис. 9.1.3) [38]. Тут точка Q одержана внаслідок повороту точки P на кут $\pi/2$ відносно осі n .

Отже, маємо

$$\begin{aligned} |v_{NQ}| &= |v_{NP}| = |v_{NP'}|, \\ v_{NP'} &= v_{NQ} \sin \theta + v_{NP} \cos \theta, \end{aligned}$$

де v_{NQ} — проекція вектора $v_{NP'}$ на вісь NQ .

Звідси одержуємо, що

$$v_{NQ} = [n * v_{NP}] = [n * [[n * v] * n]] = [n * v]$$

(див. вправу 9 в кінці параграфа). Отже,

$$\begin{aligned} v_{NP'} &= [n * v] \sin \theta + [[n * v] * n] \cos \theta = \\ &= [n * v] \sin \theta - [n * [n * v]] \cos \theta \end{aligned}$$

за властивістю антікомутативності векторного добутку. Таким чином, $TR'_\theta(v) = v_N + v_{NP'} = v + [n * [n * v]](1 - \cos \theta) + [n * v] \sin \theta$. Якщо позначити через $A_n : L3 \rightarrow L3$ перетворення, таке, що

$$A_n(v) = [n * v],$$

то $TR'_\theta = E + (1 - \cos \theta) A_n^2 + \sin \theta A_n$, де E — одинична матриця, тобто матриця A_n знаходиться з рівняння $A_n(v) = [n * v]$, де $[n * v]$ — векторний добуток векторів n і v , а A_n^2 — з рівняння

$$A_n^2(v) = [n * [n * v]].$$

Відомо, що координати векторного добутку векторів n і v , заданих своїми координатами (n_1, n_2, n_3) і (v_1, v_2, v_3) відповідно, знаходяться за допомогою визначника

$$[n * v] = \begin{vmatrix} i & j & k \\ n_1 & n_2 & n_3 \\ v_1 & v_2 & v_3 \end{vmatrix}.$$

Звідси легко побудувати матриці A_n^2 і A_n . Дійсно,

$$A_n = \begin{vmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & n_1 \\ n_2 & n_1 & 0 \end{vmatrix}, \quad A_n^2 = \begin{vmatrix} -(n_2^2 + n_3^2) & n_1 n_2 & n_1 n_3 \\ n_1 n_2 & -(n_1^2 + n_3^2) & n_2 n_3 \\ n_1 n_3 & n_2 n_3 & -(n_1^2 + n_2^2) \end{vmatrix}.$$

Наприклад, якщо $n = (1, 0, 1)$, то

$$A_n = \begin{vmatrix} 0 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \end{vmatrix}.$$

Справді,

$$A_n \cdot v = \begin{vmatrix} 0 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \end{vmatrix} \begin{vmatrix} v_1 \\ v_2 \\ v_3 \end{vmatrix} = v'.$$

Аналогічно знаходимо матрицю A_n^2 :

$$[n * v'] = \begin{vmatrix} i & j & k \\ 1 & 0 & 1 \\ -v_2 & v_1 - v_3 & v_2 \end{vmatrix} = -v_2 j + k(v_1 - v_3) - jv_2 - i(v_1 - v_3) =$$

$$=(-v_1 + v_3, -2v_2, v_1 - v_3)$$

$$A_n^2 = \begin{vmatrix} -1 & 0 & 1 \\ 0 & -2 & 0 \\ 1 & 0 & -1 \end{vmatrix}.$$

Дійсно,

$$\begin{vmatrix} -1 & 0 & 1 \\ 0 & -2 & 0 \\ 1 & 0 & -1 \end{vmatrix} \begin{vmatrix} v_1 \\ v_2 \\ v_3 \end{vmatrix} = (-v_1 + v_3, -2v_2, v_1 - v_3).$$

В результаті одержуємо

$$\begin{aligned} & \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} + \begin{vmatrix} \cos \theta - 1 & 0 & 1 - \cos \theta \\ 0 & 2 \cos \theta - 2 & 0 \\ 1 - \cos \theta & 0 & \cos \theta - 1 \end{vmatrix} + \begin{vmatrix} 0 & -\sin \theta & 0 \\ \sin \theta & 0 & -\sin \theta \\ 0 & \sin \theta & 0 \end{vmatrix} = \\ &= \begin{vmatrix} \cos \theta & -\sin \theta & 1 - \cos \theta \\ \sin \theta & 2 \cos \theta - 1 & -\sin \theta \\ 1 - \cos \theta & \sin \theta & \cos \theta \end{vmatrix}. \end{aligned}$$

Розглянемо, якою буде матриця повороту навколо осі Oz (з якої легко можна одержати матрицю перетворення повороту на кут θ в просторі L_2). В цьому випадку маємо $n = (0, 0, 1)$, для якого

$$A_n = \begin{vmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix} \quad A_n^2 = \begin{vmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{vmatrix}$$

$$\begin{aligned} TR_\theta^1 &= \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} + \begin{vmatrix} \cos \theta - 1 & 0 & 0 \\ 0 & \cos \theta - 1 & 0 \\ 0 & 0 & 0 \end{vmatrix} + \begin{vmatrix} 0 & -\sin \theta & 0 \\ \sin \theta & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix} = \\ &= \begin{vmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{vmatrix}. \end{aligned}$$

Звідси, вилучаючи в цій матриці останній рядок і стовпчик, одержуємо матрицю перетворення для повороту на кут θ в просторі L_2 , яка була побудована нами раніше:

$$TR_\theta^t = \begin{vmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{vmatrix}.$$

У загальному випадку перетворення TR_0^n відносно осі n і $TR_0^{n_i}$ відносно осі n_i некомутативні. Але якщо повороти TR_0^l і $TR_0^{l_i}$ виконуються відносно однієї і тієї ж осі, то має місце така теорема.

Теорема 9.1.6. *Перетворення повороту в просторі лінійне і ортогональне. Множина всіх поворотів відносно однієї і тієї ж осі є абелевою групою відносно операції множення поворотів.*

Доведення. Дійсно, з лінійної алгебри випливає справедливість першої частини теореми. Крім того, очевидно, що поворот відносно осі n на кут θ , а після цього поворот відносно цієї ж осі на кут θ_1 буде поворотом відносно цієї осі на кут $\theta + \theta_1$. Значить, множина таких поворотів замкнута відносно операції множення. Комутативність і асоціативність операції множення для таких поворотів досить очевидна і пропонується як вправа. Роль одиниці відіграє поворот TR_0^l на кут $\theta = 0$, а оберненим перетворенням до TR_0^l буде поворот на кут $(-\theta)$ відносно тієї ж осі n .

Практична цінність цієї теореми полягає в тому, що із властивості ортогональності перетворення TR_0^l випливає, що для того, аби знайти матрицю оберненого перетворення, необхідно лише трансформувати матрицю перетворення TR_0^l . Крім того, якщо до об'єкта застосовувати кілька поворотів навколо однієї і тієї ж осі, то порядок їх застосування може бути довільним.

В загальному випадку порядок застосування не може бути довільним.

Хоча групи переносів і поворотів навколо однієї і тієї ж осі комутативні, самі елементи цих груп не завжди можна представити між собою (див. вправу 5). З практичної точки зору це означає, що виконувати дані перетворення необхідно при строгому додержанні порядку.

Групи переносів і поворотів можна об'єднати в одну групу за допомогою операції множення, якщо покласти

$$TU'(v) = TR_0^l(v) + r = (TR_0^l, r)(v)$$

для всіх $v \in L_3$, де $r \in L_3$, а TR_0^l — поворот на кут θ відносно деякої осі l , що задана вектором n . Множення задається так:

$$(TR_0^l, r_2) * (TR_0^{l_i}, r_1) = (TR_0^l * TR_0^{l_i}, TR_0^l r_1 + r_2).$$

Теорема 9.1.7. *Множина всіх перетворень $\{TU\}$ в просторі L_3 становить групу.*

Для доведення некомутативності цієї групи зауважимо, що

$$(TR_{\theta}^l, r_1) * (TR_{\theta}^l, r_2) = (TR_{\theta}^l * TR_{\theta}^l, TR_{\theta}^l r_2 + r_1)$$

i

$$(TR_{\theta}^l, r_2) * (TR_{\theta}^l, r_1) = (TR_{\theta}^l * TR_{\theta}^l, TR_{\theta}^l r_1 + r_2)$$

не завжди рівні між собою (оскільки матриці не завжди можна переставляти).

Інші викладки досить прості.

Гвинтовим переміщенням $S'_{l, \theta}$ називається добуток перетворень повороту TR_{θ}^l і паралельного переносу T_r , при якому вектор r паралельний осі l (у цьому випадку $TR_{\theta}^l * T_r = T_r * TR_{\theta}^l$). Отже, окремим випадком гвинтового переміщення є поворот TR_{θ}^l і паралельний перенос T_r . Крім того, якщо задане деяке гвинтове переміщення, то заданий і напрямок на осі повороту — це направок вектора r .

Симетрія S_p відносно площини P — переміщення, що не змінює точки площини, а всяку іншу точку A переводить в точку A' , таку, що пряма, яка проходить через точки A і A' , перпендикулярна до площини P і точка O перетину P ділить відрізок AA' пополам.

Ковзна симетрія — це добуток $T_r * S_p = S_p * T_r$, де вектор r паралельний площині P (завдяки цьому порядок операцій несуттєвий).

Поворотна симетрія — це добуток $TR_n^l * S_p = S_p * TR_n^l$, де вісь l перпендикулярна до площини P (що теж дає комутативність операції добутку).

Виявляється, що цими прикладами перетворень вичерпуються всі переміщення простору, оскільки справедлива така теорема.

Теорема 9.1.8. *Всяке переміщення простору — це або паралельний перенос, або поворот навколо осі, або гвинтове переміщення, або симетрія відносно площини, або ковзна симетрія, або поворотна.*

Доведення цієї теореми при необхідності можна знайти в [2].

4. ГРУПИ ПЕРЕТВОРЕНЬ ПРОСТОРУ

Переміщення простору, як і переміщення площини, теж розбиваються на переміщення першого і другого роду. Паралельний перенос, поворот і гвинтове переміщення називаються **перемі-**

щеннями первого рода, а симетрія відносно площини, ковзна симетрія і поворотна — *переміщеннями другого роду*. Як для переміщень площини, так і для переміщень простору справедливе таке твердження.

Теорема 9.1.9. *Добуток двох переміщень первого рода є переміщенням первого рода, добуток переміщень первого і другого роду є переміщенням другого роду і добуток двох переміщень другого роду є переміщенням первого рода.*

Доведення пропонується як вправа.

Теорема 9.1.10. *Множина всіх переміщень простору є групою відносно операції добутку відображень.*

Доведення майже дослівно повторює доведення теореми 9.1.3.

Група всіх переміщень простору позначається $E(3)$.

Множина всіх перетворень первого рода $E_0(3)$ є підгрупою групи $E(3)$, і ця підгрупа буде інваріантною підгрупою в групі $E(3)$. Останнє випливає з теореми 9.1.10, оскільки перетворення $G * F * G^{-1}$ — перетворення первого рода. Analogічно випадку площини існує рівно два класи за цією підгрупою: вона сама і клас перетворень другого рода. Отже, $E(3)/E_0(3)$ має порядок 2 і є циклічною групою.

Розглянемо підгрупи групи $E_0(3)$. Ця група має підгрупу $T(3)$ — групу всіх паралельних переносів простору.

Теорема 9.1.11. *Група $T(3)$ — інваріантна підгрупа групи $E_0(3)$.*

Доведення аналогічне доведенню теореми 9.1.5.

Група $E_0(3)$ має нескінченне число підгруп поворотів навколо фіксованих осей.

Нехай A — деяка точка простору і $F \in E_0(3)$ — деяке перетворення первого рода. Точка A називається *нерухомою точкою перетворення* F , якщо $F(A) = A$.

Теорема 9.1.12. *Множина всіх перетворень із $E_0(3)$, які мають нерухомою точку A , є підгрупою групи $E_0(3)$.*

Доведення. Візьмемо добуток двох перетворень F_1 і F_2 , які мають точку A як нерухому. Тоді $F_1 * F_2(A) = F_2(F_1(A)) = F_2(A) = A$. Отже, множина всіх таких перетворень замкнута відносно добутку відображень. Очевидно, що $\varepsilon(A) = A$, де ε — totожне перетворення. Далі, якщо $F(A) = A$, то $F^{-1}(A) = A$, оскільки

$$A = F * F^{-1}(A) = \varepsilon(A) = F^{-1}(F(A)) = F^{-1}(A).$$

Теорема доведена.

Підгрупу всіх перетворень, які мають нерухомою точку A , позначають $SO(3)$ або $SO_A(3)$, якщо необхідно вказати точку A .

Очевидно, що $SO_A(3)$ і $SO_{A'}(3)$ ізоморфні.

Група $SO_A(3)$ включає всі групи поворотів відносно осей, які проходять через точку А.

Теорема 9.1.13. Всякий елемент $F \in SO_A(3)$ має вигляд

$$F = TR_{\theta}^{l_1} * TR_{-\theta_1}^{l_1},$$

де l і l_1 — деякі прямі, що проходять через точку А.

Доведення. Нехай B — довільна точка простору, $B \neq A$ і $B' = F(B)$. Оскільки $F \in SO_A(3)$, то $|AB| = |AB'|$. Отже, точку B' можна сумістити з точкою B за допомогою повороту $TR_{\theta_1}^{l_1}$ навколо осі l_1 , яка проходить через точку А і перпендикулярна до площини P (площина P містить точки A, B, B'), на кут θ , що дорівнює куту $B'AB$. Таким чином, добуток $TR_{\theta_1}^{l_1} * F$ залишає нерухомими точки A і B , а отже, і будь-яку точку прямої l_1 (що проходить через точки A і B). Значить, $TR_{\theta_1}^{l_1} * F$ являє собою поворот навколо осі l_1 на деякий кут α , тобто

$$TR_{\theta}^{l_1} * F = TR_{\alpha}^{l_1}.$$

Покладемо $TR_{\theta}^{l_1} = TR_{-\theta_1}^{l_1}$, тоді

$$TR_{\theta}^{l_1} * TR_{\alpha}^{l_1} = TR_{-\theta_1}^{l_1} * TR_{\alpha}^{l_1} = TR_{-\theta_1}^{l_1} * TR_{\theta}^{l_1} = TR_{\theta-\theta}^{l_1} * F = F.$$

Теорема доведена.

Нехай $F, G \in E_0(3)$ і $H = G * F * G^{-1}$. Тоді H визначається формулою

$$H * G(B) = G * F(B),$$

де B — деяка точка простору L_3 . Дійсно, оскільки G — взаємно однозначне відображення, то H буде повністю визначенім, якщо буде відомо, куди переходить точка $G(B)$. Іншими словами, відображення H буде визначене для довільної точки B , якщо відомо, куди воно переводить точку $G(B)$. Тому

$$H * G(B) = G * F * G * G^{-1}(B) = G * F(B).$$

Теорема 9.1.14. Якщо $F \in SO_A(3)$, то $H = G * F * G^{-1} \in SO_A(3)$.

Доведення. Оскільки $F(A) = A$, то

$$H * G(A) = G * F(A) = G(A),$$

тобто $G(A)$ — нерухома точка для H . Отже, $H \in SO_A(3)$.

Наслідок 9.1.1. Ніяка з груп $SO_A(3)$ не є інваріантною підгрупою в групі $E_0(3)$.

Наведені нижче перетворення не є переміщеннями, але вони теж відіграють важливу роль при перетвореннях лінійних векторних просторів.

5. МАСШТАБ

Перетворенням масштабу в просторі L_3 називається відображення, що має вигляд

$$TM(x, y, z) = (a \cdot x, b \cdot y, c \cdot z),$$

де $a, b, c > 0$, $a, b, c \in P$. В матричній формі це перетворення має вигляд

$$TM(x, y, z) = \begin{vmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{vmatrix} \begin{vmatrix} x \\ y \\ z \end{vmatrix}.$$

Множення таких перетворень виконується звичайним чином:

$$(TM_1 * TM_2)(x, y, z) = TM_1(TM_2(x, y, z))$$

або в матричній формі

$$\begin{vmatrix} a_1 & 0 & 0 \\ 0 & b_1 & 0 \\ 0 & 0 & c_1 \end{vmatrix} \begin{vmatrix} a_2 & 0 & 0 \\ 0 & b_2 & 0 \\ 0 & 0 & c_2 \end{vmatrix} \begin{vmatrix} x \\ y \\ z \end{vmatrix} = \begin{vmatrix} a_1 \cdot a_2 & 0 & 0 \\ 0 & b_1 \cdot b_2 & 0 \\ 0 & 0 & c_1 \cdot c_2 \end{vmatrix} \begin{vmatrix} x \\ y \\ z \end{vmatrix}.$$

Звідси випливає, що коли простір L_3 розглядається над деяким полем P , то операція множення перетворень масштабу комутативна і асоціативна, а саме перетворення лінійне.

Теорема 9.1.15. *Множина всіх перетворень масштабу в L_3 становить абелеву групу.*

Доведення випливає з вигляду матриці перетворення.

Перетворення масштабу не комутативне відносно перетворення переносу. Дійсно, якщо T — перенос на вектор $r_0 = (x_0, y_0, z_0)$, а TM — перетворення масштабу, таке, що $TM(v) = TM(x, y, z) = (a \cdot x, b \cdot y, c \cdot z)$, то

$$TM(T(v)) = TM(x + x_0, y + y_0, z + z_0) = \\ = (a(x + x_0), b(y + y_0), c(z + z_0)),$$

$$T(TM(v)) = T(a \cdot x, b \cdot y, c \cdot z) = (a \cdot x + x_0, b \cdot y + y_0, c \cdot z + z_0).$$

Цим некомутативність у загальному випадку встановлена. Аналогічно можна показати, що перетворення масштабу не комутативне відносно перетворення повороту.

Приклад 9.1.1

Дано дві фігури: — “зірка” і “башта, що лежить” (рис. 9.1.4). Потрібно перемістити башту в початок координат і на її шпиль прикріпити зірку.

Очевидно, що дана задача розв'язується за допомогою таких перетворень: а) перенос башти паралельно осі Ox в точку $(0, 0)$; б) поворот на -90° відносно точки $(0, 0)$; в) перенос башти паралельно осі Ox на вектор $(-1/2, 0)$; г) поворот зірки на кут -36° ; д) перенос зірки (центру) на вектор $(-1, 0)$ (рис. 9.1.4).

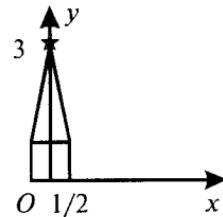
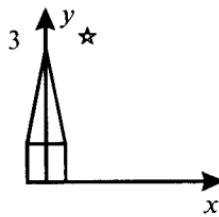
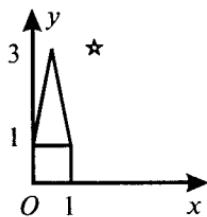
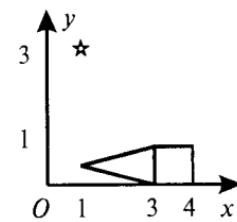


Рис. 9.1.4. Перетворення повороту

6. ПРОЕКЦІЇ

Перетворення, наведені нижче, не належать до геометричних перетворень, але вони дають можливість одержувати зображення об'єкта на екрані дисплея ЕОМ. З математичної точки зору необхідно побудувати перетворення простору L_3 на підпростір L_2 , при яких маємо необхідні графічні зображення нашого об'єкта.

Проекцією векторного простору L_3 на підпростір L_2 називається таке перетворення ПР, що $\text{ПР}^2(v) = \text{ПР}(v)$ для кожного $v \in L_3$.

Проекції бувають різні. Найпростішою проекцією є *паралельна ортогональна проекція*. Це перетворення задається так.

Нехай об'єкт описується в декартовій системі координат (x, y, z) , і в цій системі координат екран графічного дисплея відповідає прямокутнику $\{(x, y) \mid 0 \leq x \leq a, 0 \leq y \leq b\}$ на площині (x, y) . Найпростіший метод одержання образу об'єкта на екрані — це застосування перетворення $\text{ПР}(v) = v'$, де $v = (x, y, z)$, а $v' = (x, y, 0)$, до множини точок об'єкта в L_3 . Якщо всі точки образу знаходяться всередині екрана, то ми одержали зображення об'єкта, якщо ж не всі точки лежать всередині екрана, то застосовуємо перетворення масштабу, а потім перетворення ПР.

Перетворення ПР можна записати у вигляді матриці:

$$\text{ПР} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{vmatrix} \cdot \begin{vmatrix} x \\ y \\ z \end{vmatrix},$$

де $v = (x, y, z)$. Очевидно, що $\text{ПР}^2(v) = \text{ПР}(v)$ для будь-якого $v \in L_3$, і що ПР^{-1} не існує, оскільки $\det(\text{ПР}) = 0$.

Геометрична побудова ПР виконується так. Проведемо лінію проекції з точки Q , яка задає положення вектора v в просторі L_3 , в точку Q' , яка задає положення вектора v' в площині (x, y) , так щоб QQ' була ортогональна площині (x, y) (рис. 9.1.5). Ясно, що $v' = v + (z \cdot k) = (x, y, z) - (0, 0, z) = (x, y, 0) = \text{ПР}(v)$.

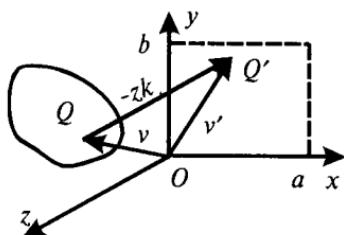


Рис. 9.1.5

Недоліком ортогональної проекції є те, що при такому перетворенні не відбивається глибина об'єкта; висота одержаного образу залишається однією й тією ж незалежно від відстані об'єкта до площини (x, y) .

Для того щоб подолати наведені недоліки, замість ліній паралельної проекції будемо будувати лінії, які виходять з однієї точки.

Знайдемо, чому в цьому випадку дорівнює $\text{ПР}(v) = v'$. Припустимо, що фіксованою точкою на осі z є точка $-z_p$ (рис. 9.1.6, 9.1.7).

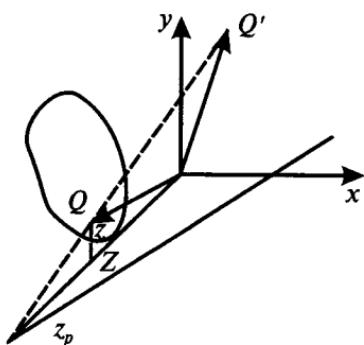


Рис. 9.1.6

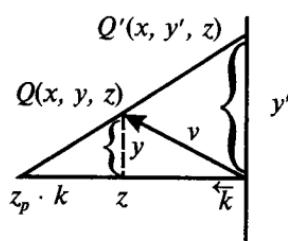


Рис. 9.1.7

З наведених рисунків одержуємо, що:

$$\frac{y'}{z_p} = \frac{y}{z_p - z} \text{ або } y' = \frac{y}{1 - z/z_p}.$$

Аналогічно дістанемо

$$x' = \frac{x}{1 - z/z_p}.$$

Отже, ПРІ має вигляд

$$\text{ПР1}(v) = \frac{1}{1 - z/z_p} (x, y, 0)$$

або

$$\text{ПР1}(v) = \frac{1}{1 - z/z_p} \text{ПР}(v).$$

Характеристику перетворень ПР і ПР1 дає таке твердження.

Твердження 9.1.3. *Перетворення ПР — лінійне, а ПР1 — нелінійне і $\text{ПР1}^2 = \text{ПР1}$.*

Д о в е д е н н я. Те, що ПР лінійне, очевидно з матриці цього перетворення. Покажемо властивості ПР1:

$$a) \text{ПР1}(a \cdot v) = \text{ПР1}(ax, ay, az) = \frac{1}{1 - az/z_p} (ax, ay, 0),$$

$$a * \text{ПР1}(v) = \frac{1}{1 - z/z_p} (ax, ay, 0) = \text{ПР1}(v);$$

$$б) \text{ПР1}(\text{ПР1}(v)) = \text{ПР1}\left(\frac{x}{1 - z/z_p}, \frac{y}{1 - z/z_p}, 0\right) =$$

$$= \frac{1}{1 - 0} \left(\frac{x}{1 - z/z_p}, \frac{y}{1 - z/z_p}, 0\right) = \text{ПР1}(x, y, z),$$

що і потрібно було довести.

§ 9.2. РЕЛЯЦІЙНА АЛГЕБРА І ПОНЯТТЯ РЕЛЯЦІЙНОЇ БАЗИ ДАНИХ

1. ОЗНАЧЕННЯ РЕЛЯЦІЙНОЇ АЛГЕБРИ

При розгляді відношень, заданих на деяких множинах, ми не брали до уваги природу цих множин і вони виступали як абстрактні математичні об'єкти. При переході до баз даних необхідно враховувати їх специфіку і властивості. Насамперед зазначимо такі факти.

Відношення R задається тільки на скінченних множинах A_1, A_2, \dots, A_n , отже, R теж скінченне. Звідси одразу випливає, що для кожного відношення R , заданого на множинах A_1, A_2, \dots, A_n , і довільної n -ки (a_1, a_2, \dots, a_n) із $A_1 \times A_2 \times \dots \times A_n$ можна визначити: хибне чи істинне відношення R для цієї n -ки. (Це можна зробити хоча б простим перебором всіх елементів із R і $A_1 \times A_2 \times \dots \times A_n$.)

При заданні відношення R переліком його елементів будемо припускати, що в цьому переліку немає однакових елементів. Таке задання відношення R називається **ненадлишковим**. Звідси випливає необхідність чистки результату операції над відношеннями (наприклад, при операції об'єднання відношень) з тим, щоб вилучити однакові елементи з одержаного відношення.

При реалізації відношень на ЕОМ необхідно дати кожному відношенню ім'я. Умовимося вважати, що ім'я відношення R ототожнюється з буквою, якою воно позначено.

Введемо поняття, які використовуватимуться надалі.

Якщо множина A_i входить в добуток $A_1 \times A_2 \times \dots \times A_n$, то вона називається **доменом**, а її індекс i — **типом** або **ім'ям** цього домену.

Якщо $(a_1, a_2, \dots, a_n) \in R \subseteq A_1 \times A_2 \times \dots \times A_n$, то елемент (a_1, a_2, \dots, a_n) називається **записом**, а його компоненти a_i — **полями** або **атрибутами**. Число атрибутів називається **степенем відношення** R . Атрибут називається **атомарним** (або **простим**), якщо він складається з одного неподільного значення. Зауважимо, що оскільки множини A_i можуть збігатися, то деякі домени можуть мати один і той же тип.

Відношення R , задане на множинах A_1, A_2, \dots, A_n , називається **нормалізованим**, якщо будь-який атрибут його запису атомарний. Всяке нормалізоване відношення R будемо представляти у вигляді таблиці з іменем R розміром $m \times n$, де m — потужність відношення R , а n — його арність. Рядки цієї таблиці являють собою записи відношення R , а стовпчики — імена атрибутів даних.

Означення 9.2.1. Сукупність записів, що задовольняють деякі нормалізовані відношення, називається (реляційною) **базою даних** (БД).

Означення 9.2.2. Нехай $\{R\}$ — сукупність всіх нормалізованих відношень, з яких складається БД. Алгебра $G = (\{R\}, W)$ називається **реляційною алгеброю**, коли сигнатура Ω містить бінарні операції об'єднання, перетину, різниці, декартового добутку, з'єднання відношень і дві унарні операції: проекції та селекції.

Перелічені операції задаються таким чином.

Проекція. Якщо R — n -арне відношення БД, то $P(R(i, j))$ означає вибір із відношення R сукупності атрибутів з іменами i та j . Аналогічно визначаються проекції $P(R(i))$, $P(R(i, j, k))$ і т.д. Інакше кажучи,

$$P(R(i, j)) = \{r(i, j) \mid r \in R\},$$

тобто по записах r відношення R формуються нові записи, що мають лише атрибути i та j .

Приклад 9.2.1

Нехай R задане таблицею R :

	A_1	A_2	A_3	A_4
x	100	5	a	
y	105	3	a	
z	501	9	a	
w	50	1	b	
w	10	2	b	
w	301	4	b	

Тоді

$$P(R(A_4)) = \begin{vmatrix} a \\ a \\ a \\ b \\ b \\ b \\ b \end{vmatrix} = \begin{vmatrix} a \\ b \end{vmatrix} \text{ — після чистки,}$$

$$P(R(A_1, A_4)) = \begin{vmatrix} x & a \\ y & a \\ z & a \\ w & b \\ w & b \\ w & b \end{vmatrix} = \begin{vmatrix} x & a \\ y & a \\ z & a \\ w & b \end{vmatrix},$$

$$P(R(A_2, A_3, A_4)) = \begin{vmatrix} 100 & 5 & a \\ 105 & 3 & a \\ 501 & 9 & a \\ 50 & 1 & b \\ 10 & 2 & b \\ 301 & 4 & b \end{vmatrix}.$$

Селекція. Нехай i, j — деякі атрибути записів відношення R , c — константа і Q — один із символів $<$, \leq , 0 , $=$, \neq , $>$, \geq . Селекція

$$S(R(i \theta c)) \text{ або } S(R(i \theta j))$$

відношення R за компонентою i та константою c або за компонентами i та j означає вибірку елементів відношення R , для яких істинне відношення $i \theta c$ або $i \theta j$. Інакше кажучи,

$$S(R(i \theta c)) = \{r \mid r \in R \text{ \& } r(i) \theta c\},$$

$$S(R(i \theta j)) = \{r \mid r \in R \text{ \& } r(i) \theta r(j)\},$$

де $\&$ — операція булевої алгебри.

Приклад 9.2.2

Нехай R — відношення, що розглядалось у прикладі 9.2.1.
Тоді

$$S(R(A_4 = a)) = \begin{vmatrix} x & 100 & 5 & a \\ y & 105 & 3 & a \\ z & 501 & 9 & a \end{vmatrix},$$

$$S(R(A_2 < A_3)) = \emptyset. \blacktriangleleft$$

Об'єднання. Операцію об'єднання відношень R і Q визначаємо, як і раніше, але з вимогою сумісності операндів за атрибутами, тобто атрибути відношень R і Q повинні бути визначені над сумісними (одного і того ж типу) доменами. Іншими словами,

$$R \cup Q = \{t \mid t \in R \vee t \in Q\},$$

де \vee — операція булевої алгебри.

Приклад 9.2.3

Нехай R — відношення, наведене в прикладі 9.2.1, а Q — відношення, задане наведеною нижче таблицею Q .

i_1	i_2
5	a
10	a
15	c
2	d
6	a
1	b

$$P(R(A_3, A_4)) \cup Q = \begin{vmatrix} 5 & a \\ 10 & a \\ 15 & c \\ 2 & d \\ 6 & a \\ 1 & b \end{vmatrix} \cup \begin{vmatrix} 5 & a \\ 3 & a \\ 9 & a \\ 1 & b \\ 2 & b \\ 4 & b \end{vmatrix} = \begin{vmatrix} 5 & a \\ 10 & a \\ 15 & c \\ 2 & d \\ 6 & a \\ 1 & b \\ 3 & a \\ 9 & a \\ 2 & b \\ 4 & b \end{vmatrix} \blacktriangleleft$$

Перетин. Перетин відношень R і Q означаємо як теоретико-множинний перетин множин R і Q , операнди яких сумісні за атрибутами. Інакше кажучи,

$$R \cap Q = \{t \mid t \in R \& t \in Q\}.$$

Приклад 9.2.4

Обчислимо $P(R(A_3, A_4)) \cap Q$, де R і Q взяті з прикладів 9.2.1 і 9.2.3:

$$P(R(A_3, A_4)) \cap Q = \left| \begin{array}{cc} 5 & a \\ 3 & a \\ 9 & a \\ 1 & b \\ 2 & b \\ 4 & b \end{array} \right| \cap \left| \begin{array}{cc} 5 & a \\ 10 & a \\ 15 & c \\ 2 & d \\ 6 & a \\ 1 & b \end{array} \right| = \left| \begin{array}{cc} 5 & a \\ 1 & b \end{array} \right|. \quad \blacktriangleleft$$

Різниця. Різниця двох відношень R і Q , операнди яких сумісні за атрибутами, визначається як і раніше, тобто

$$R \setminus Q = \{t \mid t \in R \& t \notin Q\}.$$

Приклад 9.2.5

Нехай R і Q — ті ж, що і в прикладі 9.2.4. Тоді

$$P(R(A_3, A_4)) \setminus Q = \left| \begin{array}{cc} 5 & a \\ 3 & a \\ 9 & a \\ 1 & b \\ 2 & b \\ 4 & b \end{array} \right| \setminus \left| \begin{array}{cc} 5 & a \\ 10 & a \\ 15 & c \\ 2 & d \\ 6 & a \\ 1 & b \end{array} \right| = \left| \begin{array}{cc} 3 & a \\ 9 & a \\ 2 & b \\ 4 & b \end{array} \right|. \quad \blacktriangleleft$$

Декартовий добуток. Декартовий добуток відношень R і Q — це відношення $R \times Q$, тобто звичайний декартовий добуток множин R і Q . Іншими словами,

$$R \times Q = \{rs \mid r \in R \& s \in Q\},$$

де rs — конкатенація записів r і s .

Звідси випливає, що степінь відношення $R \times Q$ дорівнює сумі степенів відношень R і Q , а потужність $R \times Q$ дорівнює $|R| \cdot |Q|$. Це свідчить про те, що отримане відношення може мати великі розміри і з цією операцією необхідно бути обережними (особливо в умовах обмежених розмірів пам'яті EOM). На практиці використовують обмежені варіанти цієї операції — різного роду операції з'єднання.

Приклад 9.2.6

Знайдемо, чому дорівнює декартовий добуток відношень $P(R(A_1, A_4))$ і $P(R(A_3, A_4)) \cap Q$ (див. приклади 9.2.1, 9.2.4).

$$P(R(A_1, A_4)) \times (P(R(A_3, A_4)) \cap Q) =$$

$$= \begin{vmatrix} x & a \\ y & a \\ z & b \\ w & b \end{vmatrix} \times \begin{vmatrix} 5 & a \\ 1 & b \end{vmatrix} = \begin{vmatrix} x & a & 5 & a \\ y & a & 5 & a \\ z & a & 5 & a \\ w & b & 5 & a \\ x & a & 1 & b \\ y & a & 1 & b \\ z & a & 1 & b \\ w & b & 1 & b \end{vmatrix}.$$

Степінь одержаного відношення дорівнює $2 + 2 = 4$, потужність його дорівнює $2 \cdot 4 = 8$. \blacktriangleleft

З'єднання або тета-з'єднання. Операція з'єднання двох відношень R і Q — це декартовий добуток відношень R і Q , на атрибути A і B яких накладена умова $A \theta B$ ($R [A \theta B] Q$), де θ — це один із символів $<$, \leq , $=$, \neq , \geq , $>$. Інакше кажучи,

$$R [A \theta B] Q = \{rs \mid r \in R \& s \in Q \& r(A) \theta s(B)\}.$$

Таким чином, в отримуване відношення попадають не всі елементи декартового добутку, а лише ті, які задовольняють відношення $A \theta B$ між атрибутами A і B .

З цього означення випливає, що атрибути A і B повинні бути сумісними.

Є кілька варіантів операції з'єднання.

Еквіз'єднання — це з'єднання відношень R і Q , де $A = B$, тобто з'єднання з перевіркою на рівність.

Натуральне з'єднання — це з'єднання, при якому атрибути з'єднання мають однакові (спільні) домени. Після з'єднання один з цих атрибутів відкидається. Отже, степінь одержаного відношення на одиницю менша суми степенів відношень операндів.

Добуток або композиція — це натуральне з'єднання, при якому відкидаються обидва атрибути з'єднання.

Приклад 9.2.7

Нехай R і Q — ті ж, що і в прикладі 9.2.4. Тоді:
з'єднання Тет:

$$\text{Tet}(R(A_3 > i_1)Q) = \left| \begin{array}{cccccc} x & 100 & 5 & a & 2 & d \\ x & 100 & 5 & a & 1 & b \\ y & 105 & 3 & a & 2 & d \\ y & 105 & 3 & a & 1 & b \\ z & 501 & 9 & a & 5 & a \\ z & 501 & 9 & a & 2 & d \\ z & 501 & 9 & a & 6 & a \\ z & 501 & 9 & a & 1 & b \\ w & 10 & 2 & b & 1 & b \\ w & 301 & 4 & b & 2 & d \\ w & 301 & 4 & b & 1 & b \end{array} \right|;$$

еквіз'єднання Eq:

$$\text{Eq}(R(A_3 = i_1)Q) = \left| \begin{array}{cccc} x & 100 & a & 5 & a \\ w & 50 & b & 1 & b \\ w & 10 & b & 2 & d \end{array} \right|.$$

композиція K :

$$K(R(A_3 = i_1)Q) = \left| \begin{array}{cccc} x & 100 & a & a \\ w & 50 & b & b \\ w & 10 & b & d \end{array} \right|.$$

Це відношення одержується із $\text{Eq}(R(A_3 = i_1)Q)$ відкиданням четвертого атрибута. ▲

2. ДЕЯКІ ВЛАСТИВОСТІ ОПЕРАЦІЙ РЕЛЯЦІЙНОЇ АЛГЕБРИ

Крім властивостей операцій перетину, об'єднання і різниці, про які йшла мова вище, операції проекції, селекції і з'єднання теж мають деякі позитивні властивості, а саме: їх можна успішно використовувати для оптимізації запитів до реляційної бази даних. Наприклад, переставляючи операції селекції, так щоб вони виконувались перед операціями з'єднання, а не після них, можна відповісти на запит швидше. Причина в тому, що операції типу з'єднання і декартового добутку працюють як генератори, виробляючи велику кількість записів. Якщо деякі операції селекції можна виконувати раніше, то число згенерованих записів можна зменшити.

Загалом слід додержуватися такого принципу: зменшувати інформацію на вході генератора записів до мінімальних розмірів.

Теорема 9.2.1. Для будь-яких відношень R і Q , сумісних за атрибутами, справедливі рівності:

- (a) $S((R \cup Q)(A \theta B)) = S(R(A \theta B)) \cup S(Q(A \theta B));$
- (б) $S((R \cap Q)(A \theta B)) = S(R(A \theta B)) \cap S(Q(A \theta B));$
- (в) $S((R \setminus Q)(A \theta B)) = S(R(A \theta B)) \setminus S(Q(A \theta B)).$

Д о в е д е н н я. Доведемо (а). $S((R \cup Q)(A \theta B)) = \{t \mid t \in R \cup Q \& t(A) \theta t(B)\} = \{t \mid (t \in R \vee t \in Q) \& t(A) \theta t(B)\}$. Використовуючи дистрибутивний закон b3) (див. булеві алгебри), можна записати:

$$\begin{aligned} &\{t \mid (t \in R \vee t \in Q) \& t(A) \theta t(B)\} = \\ &= \{t \mid (t \in R \& t(A) \theta t(B)) \vee (t \in Q) \& t(A) \theta t(B)\} = \\ &= \{t \mid t \in S(R(A \theta B)) \vee t \in S(Q(A \theta B))\} = \\ &= S(R(A \theta B)) \cup S(Q(A \theta B)). \end{aligned}$$

Аналогічно доводиться рівність (б). Її доведення пропонується як вправа.

Доведемо (в). $S(R(A \theta B)) \setminus S(Q(A \theta B)) = \{t \mid (t \in R \& t(A) \theta t(B)) \& \neg((t \in Q) \& t(A) \theta t(B))\}$. Використовуючи закони де Моргана, а потім — дистрибутивний закон, маємо

$$\begin{aligned} &\{t \mid (t \in R \& t(A) \theta t(B)) \& \neg((t \in Q) \vee \neg(t(A) \theta t(B)))\} = \\ &= \{t \mid (t \in R \& t(A) \theta t(B)) \& \neg(t \in Q)) \vee (t \in R \& t(A) \theta t(B) \& \neg(t(A) \theta t(B)))\} = \\ &= \{t \mid (t \in R \& \neg(t \in Q) \& t(A) \theta t(B)) \vee (t \in R \& 0)\} = \\ &= \{t \mid (t \in R \& \neg(t \in Q) \& t(A) \theta t(B)) \vee 0\} = \\ &= \{t \mid (t \in R \& \neg(t \in Q) \& t(A) \theta t(B))\} = S((R \setminus Q)(A \theta B)). \end{aligned}$$

Теорема 9.2.2. Для будь-яких відношень R і Q , сумісних за атрибутами, справедливі рівності:

- (а) $P((R \cup Q)(A)) = P(R(A)) \cup P(Q(A));$
- (б) $P((R \cap Q)(A)) = P(R(A)) \cup P(Q(A));$
- (в) $P((R \setminus Q)(A)) = P(R(A)) \setminus P(Q(A)).$

Д о в е д е н н я цієї теореми проводиться тим же способом, що і доведення теореми 9.2.1.

Теорема 9.2.3. Для довільних відношень R і Q , сумісних за атрибутами, справедливі рівності:

- (а) $S((R[A \theta B] Q)(C \theta D)) = (S(R(C \theta D))) [A \theta B] Q$, коли C і D — атрибути відношення R і не атрибути відношення Q ;
- (б) $S(S(R(A \theta B))(C \theta D)) = S(S(R(C \theta D))(A \theta B)) = S(R(C \theta D) \& (A \theta B)) = R(A \theta B) \cap R(C \theta D);$
- (в) $S(P(R(A_1, A_2, \dots, A_k))(A_i \theta A_j)) = P(S(R(A_i \theta A_j)) (A_1, A_2, \dots, A_k)),$ де $i, j \in \{1, 2, \dots, k\}.$

Д о в е д е н н я. Доведемо (а). $S((R[A \theta B] Q)(C \theta D)) = \{rs \mid r \in R \& s \in Q \& r(A) \theta s(B) \& rs(C) \theta rs(D)\} = \{rs \mid r \in R \& r(C) \theta r(D) \& s \in Q \& r(A) \theta s(B)\} = \{rs \mid r \in S(R(C \theta D)) \& s \in Q \& r(A) \theta s(B)\} = S(R(C \theta D)) [A \theta B] Q$ в силу комутативності булевої операції $\&$ і незалежності відношення Q від атрибутів C і D .

Доведення властивостей (б) і (в) пропонуються як вправи.

На закінчення параграфа зауважимо, що перелічені властивості операцій далеко не повні і в з'язку з важливістю оптимізації запитів до БД виникає питання про існування канонічної форми елемента реляційної алгебри. Даному питанню приділяється велика увага [55], але його обговорення виходить за рамки цієї книги. Більш детально проблематика баз даних викладена в [24, 55, 58].

Контрольні питання

1. Яке перетворення називається переміщенням?
2. Дайте означення перетворень у площині: паралельного переносу; повороту; симетрії; ковзної симетрії.
3. Дайте означення перетворень у просторі: паралельного переносу; повороту; симетрії; ковзної симетрії, гвинтової симетрії.
4. Чи буде групою множина всіх паралельних переносів площини, простору?
5. Скільки підгруп має група $E(2)$, $E(3)$?
6. Охарактеризуйте групи $T(2)$, $E_0(2)$.
7. Які Ви знаєте властивості перетворення масштабу?
8. Які Ви знаєте властивості проекцій?
9. Дайте означення реляційної алгебри.
10. Дайте означення операцій проекції; селекції; з'єднання в реляційній алгебри.

Задачі і вправи

1. Доведіть властивості ВП2 і $T_r * TR_\theta = TR_\theta * T_r$
2. Доведіть властивості ВС1 і ВС2.
3. Доведіть твердження 9.1.1.
4. Доведіть теорему 9.1.2.
5. Доведіть, що множина всіх поворотів не буде підгрупою $E_0(2)$.
Покажіть, що перетворення переносів і поворотів не завжди можна переставляти місцями.
6. Доведіть теореми 9.1.4 і 9.1.5.
7. Покажіть, що поворот на кут ψ навколо осі $n = (n_1, n_2, n_3)$, а потім поворот на кут ψ_1 навколо тієї ж осі n буде поворотом на кут $\psi + \psi_1$ навколо осі n .
8. Довести, що сукупність всіх поворотів не буде підгрупою $E_0(2)$.
9. Дайте повні доведення теорем 9.1.6 і 9.1.7.
10. Довести, що у випадку перетворень ковзної симетрії і поворотної симетрії $T_r * S_p = S_p * T_r$, $TR_\theta^l * S_p = S_p * TR_\theta^l$.
11. Доведіть теореми 9.1.9, 9.1.10 і 9.1.11.

12. Доведіть комутативність операції добутку двох поворотів відносно однієї і тієї ж осі.
 13. Дайте геометричну інтерпретацію некомутативності перетворень поворотів і переносів.
 14. Дайте повне доведення теореми 9.1.12.
 15. Покажіть, що перетворення масштабу не завжди можна переставляти з перетвореннями повороту.
 16. Дайте повні доведення теорем 9.2.1, 9.2.2 і 9.2.3.
 17. Доведіть, що для одиничного вектора n і довільного вектора v має місце тотожність $[n * [n * v] * n] = [n * v]$.
 18. Неважко помітити, що сигнатура операцій реляційної алгебри може бути скорочена, оскільки, наприклад, операцію перетину відношення можна виразити через операції об'єднання і різниці.
- Які ще з операцій реляційної алгебри можна виразити через інші операції? Побудуйте сигнатуру реляційної алгебри з мінімальним числом основних операцій.

§ 9.3. АЛГЕБРАЇЧНА СИСТЕМА СПИСКОВИХ СТРУКТУР

1. СПИСКИ. ОПЕРАЦІЇ НАД СПИСКАМИ

Нехай $F(X)$ — вільна напівгрупа з одиницею над деяким скінченним алфавітом $X = \{x_1, x_2, \dots, x_n\}$. Роль одиниці відіграє пусте слово e . Нагадаємо, що словом в алфавіті X називається довільна скінчenna послідовність символів цього алфавіту. Довільне слово $p = y_1y_2\dots y_m$ із $F(X)$ будемо називати **списком** елементів $y_1y_2\dots y_m$, а самі елементи $y_i \in X$, $i = 1, 2, \dots, m$, — складовими цього списку. При цьому елемент y_1 називається **початком**, а елемент y_m — **кінцем списку**. Якщо $p \in F(X)$, то число складових списку p називається його **довжиною** і позначається $l(p)$. Якщо p, q — два списки, то список (слово) q називається **початком (кінцем) списку** (слова) p , коли існує таке слово p' , що $p = qp'$ ($p = p'q$). Два списки $p = s_1s_2\dots s_k$ і $q = t_1t_2\dots t_l$ дорівнюють один одному, якщо $l = k$ (тобто $l(p) = l(q)$) і $s_i = t_i$, $i = 1, 2, \dots, k$.

Із розділу 2 нам відомо, що $F(X)$ є алгеброю з однією бінарною операцією конкатенації (conc) і однією нульарною операцією (пусте слово e). Розглянемо ще кілька функцій і операцій над списками, тобто над елементами множини $F(X)$.

Нехай \mathbf{N} — множина натуральних чисел і $p = y_1y_2\dots y_m$ — довільне слово із $F(X)$. Тоді

$$(1) \mathbf{head}(p) = y_1 \quad (\mathbf{head}: F(X) \rightarrow F(X)).$$

Іншими словами, функція $\mathbf{head}(p)$ дає перший символ слова p .

Безпосередньо з означення цієї функції випливають такі її властивості:

$$\begin{aligned}\text{head}(e) &= e, \\ \text{head}(y) &= y, \text{ якщо } y \in X, \\ \text{head}(\text{head}(p)) &= \text{head}(p).\end{aligned}$$

Далі:

$$(2) \text{tail}(p) = y_2 \dots y_m \quad (\text{tail}: F(X) \rightarrow F(X)).$$

Очевидно, що

$$\begin{aligned}\text{tail}(e) &= e, \\ \text{tail}(y) &= e, \text{ якщо } y \in X.\end{aligned}$$

Зміст наведених нижче функцій випливає з їх означення:

- (3) $\text{add}(p, i, x) = y_1 \dots y_i x y_{i+1} \dots y_m, 0 \leq i \leq l(p);$
- (4) $\text{sub}(p, i) = y_1 \dots y_{i-1} y_{i+1} \dots y_m, 1 \leq i \leq l(p);$
- (5) $\text{dist}(p, i) = (p_1, p_2), \text{ де } p_1 = y_1 \dots y_i, p_2 = y_{i+1} \dots y_m, 1 \leq i \leq l(p);$
- (6) $\text{hl}(p, i) = y_1 \dots y_i, 0 \leq i \leq l(p);$
- (7) $\text{tr}(p, i) = y_{i+1} \dots y_m 0 \leq i \leq l(p);$
- (8) $\text{push}(p, x) = px = \text{add}(p, l(p), x);$
- (9) $\text{pop}(p) = y_1 \dots y_{m-1} = \text{sub}(p, l(p)).$

Базовими операціями, тобто такими, через які виражаються всі інші з перелічених функцій, є операції e , conc , head і tail . Іншими словами, має місце таке просте твердження.

Теорема 9.3.1. Всі функції (3)–(9) зображаються у вигляді термів за допомогою операцій e , conc , head і tail .

Д о в е д е н н я. Розглянемо послідовно випадки:

- (3) $\text{add}(p, i, x) = \text{hl}(p, i) \text{tr}(p, i);$
- (4) $\text{sub}(p, i) = \text{hl}(p, i-1) \text{tr}(p, i);$
- (5) $\text{dist}(p, i) = (\text{hl}(p, i), \text{tr}(p, i));$
- (6) $\text{hl}(p, i) = \underbrace{\text{head}(p) \text{head}(\text{tail}(p)) \dots \text{head}(\text{tail}(\dots(\text{tail}(p)\dots)))}_{i \text{ разів}};$
- (7) $\text{tr}(p, i) = \underbrace{\text{tail}(\text{tail}(\dots(\text{tail}(p))))}_{i \text{ разів}}.$

З представлення функцій hl і tr випливає представлення функцій add і sub , а значить, і функцій push і pop .

Теорема доведена.

П р и к л а д и 9.3.1

Нехай $X = \{x, y, z, u, v, w\}$ і $p = xyzuv$. Тоді маємо:

- 1) $\text{add}(p, 3, x) = \text{head}(xyzuv) \text{head}(\text{tail}(xyzuv))$
 $\text{head}(\text{tail}(\text{tail}(xyzuv))) \text{xtail}(\text{tail}(\text{tail}(xyzuv))) =$
 $= x \text{head}(yzuv) \text{head}(\text{tail}(yzuv)) \text{xtail}(\text{tail}(yzuv)) = xy$
 $\text{head}(zuv) \text{xtail}(zuv) = xy \text{head}(zuv) \text{xtail}(zuv) = xyzxuv;$
- 2) $\text{sub}(p, 3) = \text{head}(xyzuv) \text{head}(\text{tail}(xyzuv)) \text{tail}(\text{tail}(\text{tail}(xyzuv))) =$
 $= x \text{head}(yzuv) \text{tail}(\text{tail}(yzuv)) = xy \text{tail}(zuv) = xyuv;$

- 3) $hl(p, 4) = \text{head}(xyzuv)\text{head}(\text{tail}(xyzuv))\text{head}(\text{tail}(\text{tail}(xyzuv)))$
 $= \text{xhead}(yzuv)\text{head}(\text{tail}(yzuv))\text{head}(\text{tail}(\text{tail}(yzuv))) =$
 $= xy\text{head}(zuv)\text{head}(\text{tail}(zuv)) = xyz\text{head}(uv) = xyzu;$
- 4) $tr(p, 3) = \text{tail}(\text{tail}(\text{tail}(xyzuv))) = \text{tail}(\text{tail}(yzuv)) = \text{tail}(zuv) = uv;$
- 5) $\text{dist}(p, 2) = (\text{hl}(p, 2), \text{tr}(p, 2)) = (\text{head}(xyzuv)\text{head}(\text{tail}(xyzuv)),$
 $(\text{tail}(\text{tail}(xyzuv)))) = (x\text{head}(yzuv), (\text{tail}(yzuv))) = (xy, zuv);$
- 6) $\text{push}(p, x) = \text{add}(p, l(p), x) = \text{add}(xyzuv, 5, x) = xyzuvx;$
- 7) $\text{pop}(p) = \text{sub}(p, l(p)) = \text{sub}(xyzuv, 5) = xyzu.$ ▲

Слід зазначити, що функції head і tail (push і pop) дійсно операції, в той час як решта функцій (3)–(7) не є операціями. Це дозволяє ввести таке означення.

Означення 9.3.1. Універсальна алгебра $G = (F(X), W = \{e, \text{conc}, \text{head}, \text{tail}\})$ називається **алгеброю спискових структур (ACC)**.

У цій алгебрі легко встановити справедливість таких тотожностей:

- (а) $\text{sub}(p, 1) = \text{tail}(p) = \text{tr}(p, 1);$
- (б) $\text{sub}(\text{add}(p, i, x), i + 1) = p;$
- (в) $\text{hl}(p, 0) = e \text{ i } \text{hl}(p, l(p)) = p;$
- (г) $\text{tr}(p, l(p)) = e \text{ i } \text{tr}(p, 0) = p;$
- (д) $\text{pop}(\text{push}(p, x)) = p.$

Інколи разом з функціями (3) — (9) розглядаються й інші корисні функції, такі, як:

(10) $\text{substr}(p, i, j) = x_i \dots x_{i+j-1}$, тобто це підслово слова p , яке починається з i -го символу і має довжину j ;

(11) $\text{conv}(p) = x_m x_{m-1} \dots x_2 x_1$, тобто це перестановка символів, що складає список p в зворотному порядку.

Побудова представлення функцій substr і conv у вигляді термів алгебри спискових структур пропонується як вправа.

Користуючись операціями і функціями (1)–(11) можна вводити інші функції. Розглянемо деякі з таких функцій.

Нехай алфавіт $X = \{x_1, x_2, \dots, x_n\}$ — множина, лінійно упорядкована відношенням лінійного порядку $<$ ($x_i < x_j \Leftrightarrow i < j$), $p = s_1 s_2 \dots s_k$, $q = t_1 t_2 \dots t_l$. **Лексикографічним порядком** на множині $F(X)$ називається відношення \leq , при якому $p \leq q$, якщо виконана одна з таких умов:

- 1) слово p є початком слова q (зокрема, збігається з ним);
- 2) існує таке натуральне число j , що $s_j < t_j$ і для всіх $i < j$ має місце рівність $s_i = t_i$.

Прикладом такого порядку може бути розміщення слів у словниках, якщо на відповідному алфавіті розглядати звичайний порядок букв як відношення лінійного порядку.

Відносно лексикографічного порядку множина $F(X)$ — пов-

ністю упорядкована множина, мінімальним елементом якої є пусте слово e . Користуючись цим порядком, визначимо за індукцією такі функції.

1) $l(p) : F(X) \rightarrow \mathbf{N}$ — функція довжини слова; індуктивне означення цієї функції:

$$l(e) = 0,$$

$$l(p) = 1 + l(\text{tail}(p));$$

2) $\text{subword}(p, q) : F(X) \times F(X) \rightarrow \{0, 1\}$. Ця функція дорівнює 1, якщо слово q — підслово слова p , і дорівнює 0 в протилежному випадку. Індуктивне означення цієї функції таке: для будь-якого слова q , відмінного від пустого слова e ,

$$\text{subword}(e, q) = 0,$$

$\text{subword}(p, q) =$ якщо $\text{hl}(p, l(q)) = q$, то 1, інакше
 $\text{subword}(\text{tail}(p), q)$.

Безпосередньо з означення випливають такі прості властивості: оскільки пусте слово — підслово будь-якого слова і всяке непусте слово — підслово самого себе, то

$$\text{subword}(p, e) = 1,$$

$$\text{subword}(p, p) = 1,$$

$$\text{subword}(p, q) = 0, \text{ якщо } l(p) < l(q);$$

3) $\text{subst}(p, q, r) : F(X) \times F(X) \times F(X) \rightarrow F(X)$. Результатом цієї функції є підстановка слова r замість першого входження слова q в слово p . Означення цієї функції таке: для всякого слова q , відмінного від пустого слова e , і довільного слова r

$$\text{subst}(e, q, r) = e,$$

$\text{subst}(p, q, r) =$ якщо $\text{hl}(p, (l(q))) = q$ то $r \cdot \text{tr}(p, l(q))$
 інакше $\text{head}(p) \cdot \text{subst}(\text{tail}(p), q, r)$.

Безпосередньо з означення випливають такі очевидні властивості:

$$\text{subst}(p, e, r) = rp,$$

$$\text{subst}(p, q, q) = p.$$

Приклади 9.3.2

Знайти:

a) $\text{subword}(abbcda, cd);$

б) $\text{subword}(abbc, ax);$

в) $\text{subst}(abbcda, cd, a);$

г) $\text{subst}(abcd, xa, da).$

Розв'язок:

a) $\text{subword}(abbcda, cd) = \text{subword}(bbcda, cd) =$
 $= \text{subword}(bcd, cd) = \text{subword}(cda, cd) = 1;$

б) $\text{subword}(abbc, ax) = \text{subword}(bbc, ax) = \text{subword}(bc, ax) =$
 $= \text{subword}(c, ax) = \text{subword}(e, ax) = 0;$

в) $\text{subst}(abbcda, cd, a) = a \cdot \text{subst}(bbcda, cd, a) = a \cdot b \cdot$

$\cdot substit(bcda, cd, a) = a \cdot b \cdot b \cdot substit(cda, cd, a) = abbaa;$
 г) $substit(abcd, xa, da) = a \cdot substit(bcd, xa, da) =$
 $= a \cdot b \cdot substit(cd, xa, da) = a \cdot b \cdot c \cdot substit(d, xa, da) =$
 $= a \cdot b \cdot c \cdot d \cdot substit(e, xa, da) = abcd.$ ▲

Розширимо АСС предикатом рівності $=$. Розширену таким чином АСС будемо називати *алгебраїчною системою спискових структур* (ACCC). Для цієї алгебраїчної системи справедлива така теорема.

Теорема 9.3.2. *ACCC є алгоритмічно повною системою, тобто системою, в якій можна обчислити довільну частково рекурсивну функцію.*

Д о в е д е н н я. Згідно з теоремою 3.1.3 для доведення необхідно показати, що всякий нормальній алгорифм Маркова можна представити в цій системі.

Нехай Φ — деякий нормальній алгорифм Маркова, заданий системою формул підстановки R в алфавіті X :

$$\left\{ \begin{array}{l} p_1 \rightarrow [.]q_1, \\ p_2 \rightarrow [.]q_2, \\ \dots \\ p_m \rightarrow [.]q_m \end{array} \right. \text{або} \quad \left\{ \begin{array}{l} p_1 \rightarrow [.]q'_1, \\ p_2 \rightarrow [.]q'_2, \\ \dots \\ p_m \rightarrow [.]q'_m, \end{array} \right.$$

де $q'_i = .q_i$, якщо формула підстановки заключна, і $q'_i = q_i$, якщо формула підстановки проста. Далі, нехай $p \in F(X)$ і $G = (F(X'), \{e, conc, head, tail\}, \{=\})$ — ACCC над алфавітом $X' = X \cup \{.\}$. Тоді, використовуючи операції і предикати ACCC, а також функції *subword*, *substit* і функцію довжини, можемо записати для довільного p із $F(X)$:

$$\begin{aligned}
 \Phi(p) = & \text{якщо } subword(p, p_1) = 1 \text{ то} \\
 & \text{якщо } head(q'_1) = ". \text{ то } substit(p, p_1, tail(q'_1)) \\
 & \text{інакше } \Phi(substit(p, p_1, q'_1)) \\
 & \text{інакше} \tag{9.3.1} \\
 & \text{якщо } subword(p, p_2) = 1 \text{ то} \\
 & \text{якщо } head(q'_2) = ". \text{ то } substit(p, p_2, tail(q'_2)) \\
 & \text{інакше } \Phi(substit(p, p_2, q'_2)) \\
 & \text{інакше} \dots \\
 & \text{інакше} \\
 & \text{якщо } subword(p, p_m) = 1 \text{ то} \\
 & \text{якщо } head(q'_m) = ". \text{ то } substit(p, p_m, tail(q'_m)) \\
 & \text{інакше } \Phi(substit(p, p_m, q'_m)) \\
 & \text{інакше } p.
 \end{aligned}$$

Покажемо індукцією за числом n застосованих до слова p підстановок, що функція $f(p)$, яка одержана за допомогою системи R , і функція $f'(p)$, одержана за системою (9.3.1), збігаються.

При $n = 0$ маємо $f(p) = p$, оскільки жодна із підстановок системи R не застосовна до слова p . Із (9.3.1) випливає, що $f'(p) = p$, оскільки $\text{subword}(p, p_i) = 0$. Отже, в цьому випадку $f(p) = f'(p)$.

Припустимо, що рівність виконується для всіх $m < n$, і нехай n -ю формулою підстановки є формула $p_i \rightarrow q'_i$. Можливі такі випадки: формула $p_i \rightarrow q'_i$ заключна і формула $p_i \rightarrow q'_i$ незаключна. Розглянемо ці випадки.

Нехай в першому випадку $m = n - 1$, і після виконання m -ї підстановки одержано слово p' . За припущенням індукції $f(p') = f'(p')$. З того, що $p_i \rightarrow q'_i$ застосовна до p' , випливає, що жодна з підстановок, які їй передують у системі R , не застосовні до p' , або, що те саме, $\text{subword}(p', p_j) = 0$ для всіх $j < i$, а $\text{subword}(p', p_i) = 1$ і $\text{head}(q'_i) = ". "$. Звідси випливає: $f(p') = \text{subst}(p', p_i, \text{tail}(q'_i)) = \text{subst}(p', p_i, q_i) = f'(p')$.

Другий випадок аналогічний першому, з тією лише різницею, що в першому випадку обчислення закінчуються, а в другому — продовжуються.

Теорема доведена.

Для ілюстрації роботи системи (9.3.1) розглянемо деякі приклади (див. § 3.1).

Приклади 9.3.3

1. $X = \{a, b\}$, R має вигляд

$$\begin{cases} a \rightarrow .e (= q'_1), \\ b \rightarrow b (= q'_2). \end{cases}$$

Підставивши дані формулі в систему (9.3.1), одержимо

$\Phi(p) =$ якщо $\text{subword}(p, a) = 1$ то $\text{subst}(p, a, e)$
інакше якщо $\text{subword}(p, b) = 1$ то $\Phi(p)$ інакше p .

З даного виразу випливає, що алгоритм (9.3.1) не припиняє обчислень, коли непусте слово p не має входжень букви a , тобто результат застосування (9.3.1) не буде визначений.

2. $X = \{x_1, x_2, \dots, x_n\}$, R має вигляд

$$\begin{cases} x_1 \rightarrow e, \\ x_2 \rightarrow e, \\ \dots \\ x_n \rightarrow e. \end{cases}$$

Тоді

$\Phi(p) =$ якщо $subword(p, x_1) = 1$ то $\Phi(substit(p, x_1, e))$
інакше якщо $subword(p, p_2) = 1$ то $\Phi(substit(p, x_2, e))$
інакше.....
.....
інакше якщо $subword(p, x_n) = 1$ то $\Phi(substit(p, x_n, e))$
інакше p .

3. $X = \{1, *\}, Y = \{1, *, \alpha\}$, а R має вигляд

$$\begin{cases} * \rightarrow * \\ \alpha 1 \rightarrow \alpha 1, \\ \alpha 1 \rightarrow .1, \\ e \rightarrow \alpha. \end{cases}$$

Тоді маємо

$\Phi(p) =$ якщо $subword(p, *) = 1$ то $\Phi(p)$
інакше якщо $subword(p, \alpha 1) = 1$ то $\Phi(substit(p, \alpha 1, \alpha 1))$
інакше якщо $subword(p, \alpha 1) = 1$ то $substit(p, \alpha 1, 1)$
інакше $\Phi(\alpha p)$. ▲

2. РЕАЛІЗАЦІЯ СПИСКІВ У ПАМ'ЯТІ ЕОМ

З теореми 9.3.2 випливають щонайменше два важливих висновки.

По-перше, для реалізації введених операцій і предикатів над списками в ЕОМ необхідно реалізувати лише базові операції і предикати: conc, head, tail, =.

По-друге, якщо час, необхідний на обчислення значень базових операцій, позначити відповідно t , t_1 і t_2 , то одержимо деяку міру часової складності обчислення інших функцій на ЕОМ. Дійсно, такою мірою може служити сума часових затрат на обчислення даної функції за числом базових операцій, тобто якщо m — число операцій conc, які використовуються при обчисленні даної функції, m_1 — число операцій head, m_2 — число операцій tail, то часова складність обчислення всієї функції в цілому буде сумою $m \cdot t + m_1 \cdot t_1 + m_2 \cdot t_2$.

Приклади 9.3.4

Нехай $p = xyzuv$. Тоді маємо:

$$1) add(p, 3, x) = head(xyzuv)head(tail(xyzuv))$$

$$head(tail(tail(xyzuv))))xtail(tail(tail(xyzuv))))$$

значить, часова складність обчислення функції $add(p, 3, x)$ дорівнює $4 \cdot t + 3 \cdot t_1 + 6 \cdot t_2$.

2) $\text{sub}(p, 3) = \text{head}(xyzuv)\text{head}(\text{tail}(xyzuv))\text{tail}(\text{tail}(\text{tail}(xyzuv)))$,
 значить, часова складність обчислення функції $\text{sub}(p, 3)$ дорівнює
 $2 \cdot t + 2 \cdot t_1 + 4 \cdot t_2$.

Нехай маємо список: $p = \text{елем1 елем2 елем3 елем4}$. Тоді його можна записати таким чином:

$|\text{елем1}| \text{наст} | \rightarrow |\text{елем2}| \text{наст} | \rightarrow |\text{елем3}| \text{наст} | \rightarrow |\text{елем4}| 0 |$.

Реалізація цього списку в пам'яті ЕОМ виконується за допомогою двох масивів: ЕЛЕМЕНТ і НАСТУПНИЙ (рис.9.3.1, 9.3.2).

ЕЛЕМЕНТ	НАСТУПНИЙ
0	—
1	1
2	елем1
3	3
4	елем4
5	0
6	елем2
7	4
8	елем3
9	2
...	...

Рис. 9.3.1

ПОПЕРЕДНІЙ	ЕЛЕМЕНТ	НАСТУПНИЙ
0	—	—
1	—	1
2	0	елем1
3	елем1	3
4	3	елем4
5	елем4	0
6	0	елем2
7	елем2	4
8	4	елем3
9	елем3	2
...

Рис. 9.3.2

Масив — це структура даних прямого доступу, тобто така структура, доступ до елементів якої виконується за обмежених параметрами ЕОМ час, якщо відомий індекс (номер) елемента.

Зауважимо, що елементи списку можуть іти не підряд в масиві, який зображає цей список, а довільно. Наприклад, в масиві на рис. 9.3.1 елемент 2 має індекс 4, а елемент 4 — індекс 2 тощо. Таке заповнення масиву має як свої переваги, так і недоліки.

3. СКЛАДНІСТЬ ВИКОНАННЯ ОПЕРАЦІЙ НАД СПИСКАМИ

Як уже говорилось, для реалізації операцій над списками необхідно спочатку реалізувати операції conc, head і tail. З представлення списку в машині випливає, що операції head і tail природно реалізуються за допомогою відповідних покажчиків. Розглянемо інші операції.

Якщо елементи списку разміщені в масиві, який його представляє, підряд, то такі операції, як add і sub реалізуються, очевидно, за обмежений час. Якщо ж вони розміщені не підряд, то

необхідно ще знайти індекс потрібного елемента в масиві. Наприклад, при обчисленні функцій $\text{add}(p, i, x)$ і $\text{sub}(p, i)$ необхідно затратити ще i одиниць часу для пошуку i -го елемента в масиві.

Таким чином, має місце таке твердження.

Твердження 9.3.1. *Операції add і sub виконуються за обмежений час, якщо відоме місце знаходження елемента в списку.*

Аналогічним чином можна переконатися в справедливості і такого твердження.

Твердження 9.3.2. *Операції conc і dist виконуються за обмежений час, якщо відомі індекс останнього компонента i і індекс компонента, який безпосередньо передує місцю розщеплення списку, відповідно.*

4. РІЗНОВИДИ СПИСКІВ

Виділяють деякі види списків, які розрізняються за операціями, що виконуються над ними.

Стеком, або **магазином**, називається список, в якому елементи додаються і вилучаються лише на кінці цього списку.

Стек — це список, який реалізується одним масивом ЕЛЕМЕНТ (див. рис. 9.3.1) і одним покажчиком, який вказує на останній елемент в списку. Для стека має місце правило:

останнім прийшов — першим вийшов.

Операції над стеком:

- 1) занести елемент в стек (push);
- 2) забрати елемент із стека (pop).

Очевидно, що операції push і pop не залежать від числа елементів у стеку і тому виконуються за обмежений час.

Черга — це список, у якого елементи додаються з одного (переднього) кінця, а вилучаються з другого.

Черга реалізується одним масивом ЕЛЕМЕНТ і двома покажчиками на перший і останній елементи в цьому списку.

Для черги виконується таке правило:

першим прийшов — першим вийшов.

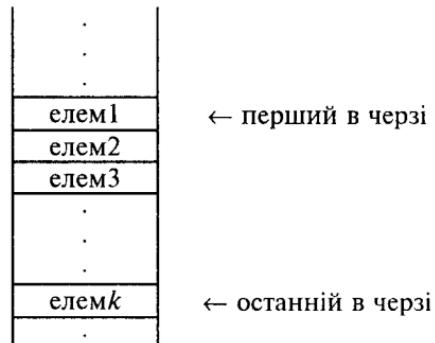


Рис. 9.3.3

Двосторонній список — це список, в якого кожний компонент зв'язаний з попереднім і наступним. Такий список реалізується за допомогою трьох масивів: ЕЛЕМЕНТ, НАСТУПНИЙ і ПО-ПЕРЕДНІЙ (див. рис. 9.3.2).

Твердження 9.3.3 (про властивості двосторонніх списків). Виконання операцій add і sub над елементами двосторонніх списків не залежить від місцезнаходження попереднього елемента і реалізується за обмежений час.

5. ЗАДАННЯ ГРАФІВ ЗА ДОПОМОГОЮ СПИСКІВ

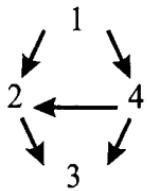
Як видно з попереднього розгляду, матриці суміжностей — досить природний спосіб задання графа в пам'яті ЕОМ. Якщо граф має n вершин, то для його задання необхідно мати масив розміру $n \times n$, тобто для задання матриці суміжностей графа необхідно n^2 елементів пам'яті. У тому випадку, коли матриця симетрична або симетрична і головна її діагональ нульова, то це скорочує необхідний об'єм пам'яті до $1/2 \cdot n \cdot (n - 1)$ елементів. Часто буває так, що багато елементів матриці суміжностей нульові і більша частина зайденої пам'яті стає зайвою. Незважаючи на це, багато задач на графах при такому заданні розв'язується досить просто і ефективно. Але існує цілий ряд задач, які розв'язуються більш ефективно, якщо задавати графи в пам'яті ЕОМ іншими способами, наприклад списками.

Задання графа у вигляді списків можна виконати різними способами. Один з них — задання графа списками суміжностей. У цьому випадку зожною вершиною графа і зв'язується список вершин $L(u)$, суміжних з цією вершиною (рис. 9.3.4, в).

Другий спосіб спискового задання графа — це таблиця списків суміжностей (рис. 9.3.4, г).

Вибір того чи іншого способу задання графа залежить від алгоритма розв'язку задачі і тому кожне задання має свої переваги і свої недоліки.

Приклад 9.3.5



а

	1	2	3	4
1	0	1	0	1
2	0	0	1	0
3	0	0	0	0
4	0	1	1	0

б

Рис. 9.3.4

1⇒	2		→	4	0
2⇒	3	0			
3⇒	0	0			
4⇒	2		→	3	0

в

1		5
2		7
3		0
4		8
5	2	6
6	4	0
7	3	0
8	2	9
9	3	0

ε

Орієнтований граф G (див. рис. 9.3.4, а) заданий у вигляді: б) матриці суміжностей; в) списку суміжностей; г) таблиці списку суміжностей.

Задання графів матрицями суміжностей має свої переваги і недоліки.

Переваги. 1. Якщо граф G представляється матрицею суміжностей, то наявність або відсутність ребра (u , v) в графі G визначається за обмежений час.

2. Пам'ять, яку займає граф G , пропорційна $|V|^2$, навіть коли G має тільки $O(|V|)$ ребер. Часто рядки і (або) стовпці матриці суміжностей зображуються двійковими векторами. Це представлення може значно підвищити ефективність алгоритмів на графах.

Недоліки. Пам'ять, яку займає граф G , пропорціональна $|V| + |E|$, де E — множина ребер. Це представлення часто використовується, коли $|E| \ll |V|$.

6. ЗАДАННЯ МНОЖИН І ТЕРМІВ ЗА ДОПОМОГОЮ СПИСКІВ

Множини здебільшого задаються списками, характеристичними векторами, графами, ациклічними графами, деревами.

Твердження 9.3.4. Якщо множини A і B не перетинаються, то операція об'єднання цих множин виконується за обмежений час, тобто не залежить від їх розмірів.

Д о в е д е н я. Представимо множини A і B у вигляді списків. Тоді можна сказати, що об'єднання цих множин зводиться до операції конкатенації цих двох списків. За твердженням 9.3.2 наше твердження має місце.

Спискове задання множин має такі недоліки:

1) операції перетину і об'єднання множин A і B вимагають часу, пропорціонального сумі розмірів множин A і B ;

2) пошук елемента в множині A вимагає часу, пропорціонального розміру множини A .

Альтернативою до спискового задання множин є задання множин характеристичними векторами.

Твердження 9.3.5. Якщо U — універсальна множина (тобто всі множини, що розглядаються, є підмножинами множини U), така, що $|U| = n$, і її підмножини A і B задаються характеристичними векторами, то операції об'єднання і перетину вимагають n бітових

операций, а операція ЗНАЙТИ(a, A) не залежить від розмірів множини A .

Доведення. Задаючи множини A і B характеристичними векторами довжини n , зводимо операції $A \cup B$ і $A \cap B$ до операцій кон'юнкції та диз'юнкції над відповідними характеристичними векторами цих множин, а операція ЗНАЙТИ(a, A) полягає в тому, щоб впевнитися, що $v(a)$ дорівнює 1 або 0.

Задання множин характеристикими векторами має такі недоліки:

1) всі операції $A \cup B$ і $A \cap B$ вимагають часу, пропорціонального $n = |U|$, а не $|A|$ і $|B|$;

2) пам'ять, необхідна для задання множин A і B , теж пропорціональна n , а не $|A|$ і $|B|$.

Вибір того чи іншого представлення множини залежить від конкретної задачі.

На завершення даного розділу зробимо ще деякі зауваження відносно часової складності алгоритмів і програм, оскільки це одна з основних характеристик при оцінці їх якості.

В теорії складності обчислень [4] з кожною конкретною задачею (а точніше, з алгоритмом розв'язання даної задачі) пов'язують деяке число, яке називається її *розміром* і яке виражає міру кількості вхідних даних. Наприклад, розміром задачі множення двох цілих чисел може служити кількість розрядів у двійковому записі цих чисел; розміром задачі на графі може служити число ребер або число вершин у даному графі і т.д.

Час, що витрачається алгоритмом на обчислення результату, як функція від розміру задачі називається *часовою складністю алгоритму*. Якщо ця функція є константою (тобто не залежить від розміру задачі), то говорять, що алгоритм виконується за постійний час або за константу. Якщо ця функція виражається поліномом, то говорять, що алгоритм має поліноміальну оцінку часової складності. Якщо функція показникова, то говорять, що алгоритм має експоненціальну оцінку часової складності і т.д.

Якщо алгоритм реалізується програмою в деякій мові програмування, то його часову складність можна визначити більш точно, оскільки точніше можна вимірити складність виконання кожного оператора цієї мови. Існує кілька критеріїв оцінки операторів мови програмування, але найбільш поширеним є *рівномірний ваговий критерій*. За цим критерієм на виконання кожного оператора витрачається одна одиниця часу.

Приклад 9.3.6

Алгоритм множення цілих невід'ємних чисел x і y , $x, y \in \mathbb{N}$.

початок.

u: = *x*;

v: = *y*;

z: = 0;

поки *u* ≠ 0 виконувати

(*u*: = *u* – 1;

z: = *z* + *v*);

кіц;

друкувати (*z*);

зупинитись;

кінець.

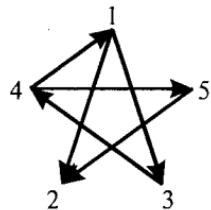
Якщо за розмір задачі взяти число одиниць, з яких складається число *x*, то число виконаних операцій додавання і віднімання, необхідних для обчислення добутку чисел *x* і *y*, може бути часовою мірою складності алгоритму MN, яка виражається функцією $3 + 3 \cdot x + 2 = O(x)$. ▲

Контрольні питання

1. Що називається списком?
2. Які операції над списками виконуються в АСС?
3. Дайте означення АСС.
4. Дайте означення АССС.
5. Які переваги і недоліки існують при заданні множин, термів, графів за допомогою списків?
6. Що називається матрицею суміжностей графа?
7. Що називається характеристичним вектором скінченної множини?
8. Які ви знаєте різновиди списків?

Задачі і вправи

1. Побудуйте терми для функцій substr і conv в алгебрі спискових структур.
2. Доведіть твердження 9.3.1 і 9.3.2.
3. Побудуйте алгоритм обходу складеного об'єкта для внутрішнього порядку.
4. Побудуйте алгоритм обходу складеного об'єкта для оберненого порядку.
5. Побудуйте матрицю суміжностей, списки суміжностей і таблицю списків суміжностей для графа



6. Доведіть, що множина слів $F(X)$ в деякому скінченному алфавіті X буде повністю упорядкованою відносно лексикографічного порядку.

7. Дайте індуктивне означення функції $L(P) = \sum_{i=1}^k l(p_i)$ — сумарної довжини слів множини $P = \{p_1, p_2, \dots, p_k\}$.

P o z d i l 10

ТЕОРІЯ АВТОМАТІВ І ГРАФІВ У КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЯХ

У даному розділі описується застосування теорії автоматів, які відрізняються від тих, про які йшла мова в розділах 6, 7. Теорія автоматів широко використовується в сучасних інформаційних технологіях, таких, як системи підготовки та редакції текстів, системи комп'ютерної алгебри, системи штучного інтелекту тощо. Висвітлюються основні моменти цих застосувань і наводяться відповідні приклади, які демонструють принципи побудови тих чи інших систем.

§ 10.1. УНІВЕРСАЛЬНИЙ ПРОГРАМНИЙ АВТОМАТ

Одне із найвагоміших досягнень сучасної техніки — це побудова універсальних програмних автоматів, тобто таких автоматичних перетворювачів інформації, які дають можливість реалізовувати довільні алгоритми. Такими автоматами є сучасні універсальні електронні обчислювальні машини (ЕОМ). Термін *універсальні* щодо цих машин розуміють як універсальність по відношенню до обчислювальних алгоритмів.

Оскільки всякий алгоритм може бути зведенний, як про це говорилося вище, до обчислення значення деякої частково рекурсивної функції, то універсальність по відношенню до обчислювальних алгоритмів є універсальністю взагалі. З'ясуємо питання про те, що повинна вміти робити система забезпечення роботи ЕОМ, щоб гарантувати цю універсальність.

Принцип керування, який забезпечує алгоритмічну універсальність ЕОМ, являє собою розвиток і узагальнення принципу, покладеного в основу алгоритмічної системи Поста. Як і в схемі

Поста, інформація в ЕОМ зберігається в пам'яті, яка розбита на чарунки (чарунки пам'яті). Правда, на відміну від машин Поста в кожній чарунці пам'яті може зберігатися не одне із чисел 0 чи 1, а ціле слово в алфавіті {0, 1}, яке може складатися із значної (як правило, 30—40) кількості двійкових цифр 0 та 1. У зв'язку з цим розглядають вмістожної чарунки пам'яті не як одну букву алфавіту, а як слово (інформаційне слово) в двійковому алфавіті {0, 1}. Двійкові цифри, які входять у запис слова, називаються *двійковими розрядами*, а саме слово — *двійковим кодом*, або просто *двійковим числом*.

Крім заміни двійкових цифр багаторозрядними двійковими кодами, існує ще одна істотна відмінність в організації пам'яті універсальної ЕОМ (ҮЕОМ) і пам'яті для алгоритмів Поста. Як абстрактна алгоритмічна схема, машина Поста є схемою з необмеженим об'ємом пам'яті, в той час як реальні технічні пристрої, такі, як ҮЕОМ, мають обмежені об'єми пам'яті. Маючи на увазі те, що нескінченну пам'ять неможливо реалізувати в жодному технічному пристрої, прийнято називати машину *універсальною*, якщо організація її керування і набір операцій такі, що дають можливість реалізувати будь-який алгоритм при умові необмеженого об'єму пам'яті.

Універсальність сучасних ЕОМ на практиці забезпечується тим, що крім швидкодіючої (оперативної) пам'яті, ЕОМ забезпечується відносно повільними (зовнішніми) пристроями пам'яті, які можуть обмінюватися інформацією з оперативним пристроям пам'яті. Об'єм зовнішньої пам'яті можна вважати майже необмеженим, що і забезпечує (при наявності засобів обміну кодами з оперативною пам'яттю) практичну можливість реалізації будь-якого алгоритму.

Послідовність операцій, що їх виконує ҮЕОМ, задається за допомогою програми, яка зкладається в пам'ять ЕОМ і являє собою упорядковану скінченну множину наказів (команд). Ці накази можна розглядати як узагальнення наказів, які використовуються при побудові алгоритмічної системи Поста. На відміну від схеми Поста, в якій при виконанні кожного наступного наказу активна чарунка зміщується не більше ніж на один крок ліворуч чи праворуч, в ҮЕОМ передбачена можливість довільної зміни положення активної чарунки від наказу до наказу. Для цього в кожний наказ вводиться номер однієї чи кількох чарунок пам'яті, які стають активними при виконанні даного наказу. Ці номери чарунок пам'яті в ЕОМ називають *адресами*. Число адрес, які можна використовувати в наказах сучасних ЕОМ, як правило, лежить у межах від 1 до 4. Відповідно до цього розрізняються одно-, двох-, трьох- і чотирьохадресні накази.

Розглянемо спочатку ЕОМ з чотирьохадресною системою організації наказів, тобто ЕОМ, в яких максимальне число адрес в наказі дорівнює чотирьом. Різні *типи* наказів відповідають різним операціям, що виконуються ЕОМ. Накази в ЕОМ записуються у вигляді двійкових чисел (як правило), які можуть зберігатися як в оперативній, так і в зовнішній пам'яті ЕОМ.

Будемо вважати, що в кожній чарунці пам'яті може зберігатися або командне слово (наказ), або інформаційне слово. Командне слово поділяється на *операційну частину* та *адресну частину*. В першій частині стоїть код операції, яка виконується під час дії наказу, що зображається цим командним словом. В другій частині розміщаються адреси чарунок, активних під час дії наказу.

Операції, які виконує ЕОМ, поділяються на кілька класів. До першого класу належать *арифметичні* операції — додавання, віднімання, множення та ділення. Чотирьохадресні накази для арифметичних операцій мають, як правило, таку структуру: в операційній частині наказу стоїть код операції — її умовний номер, який означає ту чи іншу дію (наприклад, 1 — додавання, 2 — віднімання, 3 — множення, 4 — ділення). Перші дві адреси в адресній частині наказу використовуються для задання адрес аргументів операції, яка вказана в наказі. Третя адреса в наказі служить для зберігання результату виконання даної арифметичної операції. Нарешті, четверта використовується для задання адреси чарунки пам'яті, в якій зберігається наказ, що повинен виконуватися після діючого наказу.

Аналогічно будуються накази для виконання *логічних операцій*. Логічні операції в більшості своїй бінарні і виконуються по-порозрядно, тобто окремо дляожної пари відповідних двійкових розрядів аргументів операції. До таких операцій належать порозрядна кон'юнкція і порозрядна диз'юнкція. Існують також і унарні логічні операції. До них належать ліві і праві зсуви, які перетворюють даний двійковий код $x_0...x_n$ на код $x_1...x_n 0$ і $0 x_0...x_n$ відповідно.

Особливу роль відіграють так звані *операції передачі керування*, які служать для зміни порядку виконання наказів програми залежно від результатів, що одержуються в процесі її виконання. Типовою операцією передачі керування (яку часто називають операцією *умовного переходу*) є *операція умовного переходу за точним збігом слів*. Перші дві адреси в наказі, що реалізує дану операцію, використовуються для чарунок пам'яті, з яких вибираються слова, що порівнюються між собою. Якщо слова збігаються (рівні між собою), наступний наказ береться з чарунки, яка міститься за третьою адресою, якщо ж слова не збігаються — за

четвертою адресою в наказі умовного переходу. Можливі й інші умовні переходи, наприклад умовний переход залежно від знаку різниці двох слів або залежно від знаку одного якого-небудь слова (в останньому випадку очевидно, що в наказі умовного переходу досить мати місце для трьох, а не для чотирьох адрес).

Пам'ять ЕОМ влаштована таким чином, що при читанні слова з якої-небудь чарунки пам'яті для виконання тієї чи іншої операції виникає копія цього слова, яка надходить у відповідний пристрій для виконання операції, а саме слово залишається в чарунці, з якої воно було прочитане (скопійоване). Якщо виконується запис нового слова в ту чи іншу чарунку пам'яті, то інформація, яка була в цій чарунці до запису, автоматично знищується (стирається).

Беручи до уваги всі ці властивості пам'яті, неважко помітити, що для виконання довільного алгоритму Поста достатньо користуватися лише операцією (алгебраїчного) додавання, однією з операцій умовного переходу за точним збігом слів та операцією зупинки машини.

Дійсно, умовимося оперувати лише двома будь-якими інформаційними словами — p і q , перше з яких ототожнимо з нулем, а друге — з одиницею двійкового алфавіту будь-якого заданого алгоритму Поста A . Пам'ять УЕОМ M роздіlimо на три частини. Перша складається всього із п'яти чарунок: a_{-1} , a_0 , a_1 , b_0 і b_1 , в яких розміщені числа -1 , 0 , 1 , p , q ; в чарунках другої частини розмістимо програму алгоритму A ; нарешті, третя частина імітує інформаційну стрічку алгоритму A .

Перейдемо до безпосереднього моделювання наказів алгоритму Поста A наказами машини M .

Як нам відомо з § 3.1, алгоритми Поста можуть використовувати шість різних типів наказів. Наказ шостого типу безпосередньо моделюється відповідним наказом машини M . Накази перших двох типів (запис у чарунку числа 0 або 1) моделюються наказами машини M , які виконують пересилання інформації із чарунок b_0 чи b_1 в активну чарунку, вказану, наприклад, в третьій адресі машинного наказу. Зрозуміло, що таке пересилання можна виконати наказом додавання, в перших двох адресах якого стоїть пара адрес чарунок a_0 і b_0 або a_0 і b_1 , а в третьій — адреса активної чарунки (четверта адреса, як і в машині Поста, використовується для розміщення адреси наказу, який повинен виконуватися наступним після діючого).

Неважко помітити, що постовський наказ п'ятого типу моделюється машинним наказом умовного переходу за точним збігом слів. Для цього достатньо порівняти слово в чарунці b_1 зі словом

в активній чарунці і перейти залежно від результатів цього порівняння до одного з двох наказів.

Нарешті, всякий постовський наказ третього чи четвертого типів моделюється за допомогою групи машинних наказів, число яких дорівнює числу наказів першого, другого і п'ятого типів у заданій програмі алгоритму Поста *A*.

Дійсно, нехай r' — довільний наказ першого, другого чи п'ятого типу алгоритму *A*. Згідно із сказаним вище він моделюється одним машинним наказом, який позначимо r . Нехай постовський наказ q' зміщує активну чарунку на одну одиницю праворуч. Для машинної програми таке зміщення може бути виконане лише за рахунок зміни на $+1$ всіх адрес активних чарунков. Такі чарунки зустрічаються лише в наказах машини, яка моделює постовські накази першого, другого та п'ятого типів. За рахунок підходящеї нумерації адрес, не обмежуючи загальності, можна допустити, що адреси активних постовських чарунок записуються в будь-якій заданій, наприклад останній, адресі в наказі. Якщо вважати адреси цілими двійковими числами, то приходимо до висновку, що для зміни адреси активної чарунки в наказі r на $+1$ достатньо додати до її адреси в команді r число 1, розміщене у відповідній адресі чарунки a_1 . Аналогічно виконується зміщення ліворуч.

Із сказаного зрозуміло, що алгоритмічна універсальність будь-якого цифрового автомата, який працює під керуванням програми, буде забезпечена, якщо за допомогою операцій, які виконуються цим автоматом, можна виконати такі чотири операції:

- 1) операцію пересилання вмісту будь-якої чарунки пам'яті в будь-яку іншу чарунку пам'яті;
- 2) операцію додавання констант до командного слова (наказу), розміщеного в будь-якій чарунці пам'яті, з метою зміни величини заданої адреси в наказі на 1 або -1 ;
- 3) операцію умовного переходу за точним збігом слів;
- 4) операцію (безумовної) зупинки машини.

Очевидно також, що, крім перелічених операцій, в наборі кожної ЕОМ мають бути також операції введення (виведення) інформації в машину (із машини).

Звідси випливають також умови функціональної повноти, які повинна задовольняти універсальна мова програмування.

Розглянемо тепер питання про шляхи зменшення кількості адрес в наказах. Перш за все неважко зрозуміти, що четверта адреса в наказі r може бути вилучена за рахунок такого слідування наказів в пам'яті ЕОМ, щоб адреса наказу в r , який повинен виконуватися наступним, була завжди на одиницю більшою

адреси самого наказу p . Інакше кажучи, використання четвертої адреси стає непотрібним при умові, що порядок слідування наказів в чарунках пам'яті відповідає порядку їх виконання машинною.

Зміна послідовного порядку виконання наказів можлива у випадку, коли виконується наказ умовного переходу. Як зазначалося вище, в чотирьохадресній системі умовних наказів одна з адрес використовується для задання наступного наказу при невиконанні заданої умови, а інша — для задання наступного наказу при виконанні цієї умови. При заміні чотирьохадресної системи наказів трьохадресною приймають рішення, що в першому випадку (при невиконанні умови) після наказу умовного переходу виконується наказ, записаний у наступній по порядку чарунці пам'яті, і лише в другому випадку, тобто при виконанні умови, одна з адрес в умовному наказі використовується для задання адреси наказу, який повинен виконуватися наступним.

Із сказаного випливає, що четверта адреса в наказі може стати зайвою не тільки в звичайних наказах, які не змінюють подальшого порядку виконання наказів, але і в наказах умовних переходів. Отже, ми приходимо до *трьохадресної системи наказів*.

Подальше зменшення числа адрес у наказах може бути досягнуто за рахунок фіксації деякої допоміжної чарунки пам'яті, конструктивно відокремленої від інших чарунк ЕОМ. Після фіксації такої чарунки відкривається простий шлях, який дає змогу зменшити число адрес у наказі до мінімуму, тобто до однієї адреси. Пояснимо це на прикладі операції додавання, яка вимагає трьох адрес: першого аргументу a , другого аргументу b і адреси c для зберігання суми. За допомогою фікованої чарунки d цю трьохадресну операцію можна виконати шляхом послідовного виконання трьох одноадресних операцій — пересилання числа з a у фіковану чарунку d , додавання числа, яке міститься в чарунці b , до числа в чарунці d і, нарешті, пересилання числа із чарунки d в чарунку з адресою c . Оскільки при цьому можуть змінюватися лише адреси a , b , c , а адреса d раз і назавжди фікована, то всі три вказані операції реалізуються за допомогою одноадресних наказів.

Описаний спосіб приводить нас до *одноадресної системи наказів*. Маючи одноадресну систему наказів, неважко побудувати також *двохадресну систему наказів*. Друга адреса при цьому може використовуватися як для задання адреси наступного наказу, так і для задання числових кодів (інформаційних слів), над якими виконуються операції.

Всякий пристрій, який має розбиту на окремі чарунки дискретну пам'ять і роботою якого можна керувати за допомогою

послідовності командних слів — наказів, що розміщені в цих чарунках, називається *програмним автоматом*, а сама послідовність наказів — *програмою роботи* такого автомата. Якщо набір операцій (типів наказів) програмного автомата функціонально повний для реалізації будь-якого алгоритму, то такий автомат називається *універсальним програмним автоматом*.

Ми розглянули одне досить важливе теоретичне застосування нескінчених автоматів для встановлення функціональної повноти набору операцій програмного автомата, а тепер перейдемо до розгляду деяких практичних застосувань скінчених автоматів.

§ 10.2. ЗАСТОСУВАННЯ В СИСТЕМАХ ПІДГОТОВКИ ТА РЕДАКЦІЇ ТЕКСТІВ

1. ІДЕНТИФІКАЦІЯ СЛІВ ПРИ ПОБУДОВІ ТЕКСТОВИХ РЕДАКТОРІВ

Розглянемо вільну напівгрупу $F(X)$ над деяким скінченним алфавітом $X = \{x_1, x_2, \dots, x_m\}$. Нехай $p, q \in F(X)$ — довільні слова, де $l(p) < l(q)$. Необхідно знайти одне (перше) входження слова p в слово q або всі входження слова p в слово q . Задачі такого типу носять назву *задач ідентифікації* (у даному випадку задачі ідентифікації слів). Задачі ідентифікації — складова частина проблем, пов'язаних із редагуванням текстів, пошуком даних у базах даних і символічними обчислennями. Розглянемо проблему пошуку слова p в слові q , одну з основних при розв'язуванні перелічених задач.

Оскільки слово p задане, а слово q може бути довільним, то задача ідентифікації слова p в слові q зводиться до побудови скінченного автомата A , який представляє регулярну мову, що має вигляд $L = \{X\} \cdot p = \{x_1 \vee x_2 \vee \dots \vee x_m\} \cdot p$. Очевидно, що дана мова регулярна, і тоді за теоремою синтезу існує автомат A , який представляє цю мову деякою множиною станів F . Зауважимо, що $p \in L$, оскільки $e \in \{X\}$. Більше того, слово p — найкоротше слово із L , яке необхідно знайти в слові q .

Побудова автомата A за словом p виконується за допомогою функції “відмов” g [4]. Ця функція визначається так. Нехай $p = b_1 b_2 \dots b_k$ і b_j — j -й символ слова p . Тоді $g(j)$ — найбільше ціле число $s < j$, для якого $b_1 b_2 \dots b_s = b_{j-s+1} \dots b_j$. Якщо такого s не існує, то $g(j) = 0$.

Приклад 10.2.1

Нехай $X = \{a, b\}$ і $p = ababba$. Функція g набуває таких значень:

j	1	2	3	4	5	6
$g(i)$	0	0	1	2	0	1

Наприклад, $g(4) = 2$, тому що ab — найбільше початкове слово, яке збігається з кінцевим словом слова $abab$ ($l(abab) = 4$). \blacktriangleleft

Нагадаємо, що коли слово p представляється у вигляді добутку $p = xyz$, де x, y, z — деякі слова, то слово x називається **початком слова** p , або **префіксом**, слово y — **підсловом** слова p , а слово z — **суфіксом**.

Алгоритм обчислення значень функції відмов g для заданого слова p має такий вигляд.

ВІДМОВА(p)

початок

1. $g(1) := 0$;
2. Для $j = 2$ до $l(p)$ виконати
 3. $i := g(j - 1)$;
 4. Поки $((x_j \neq x_{j+1}) \& (i > 0))$ виконати
 5. $i := g(i)$;
 6. КІЦ;
 6. Якщо $((x_j \neq x_{j+1}) \& (i = 0))$ то $g(j) := 0$
інакше $g(j) := i + 1$;

КІЦ;

кінець

Характеристику алгоритму ВІДМОВА дають такі твердження.

Теорема 10.2.1. Алгоритм ВІДМОВА правильно обчислює значення функції g .

Доведення ведеться індукцією за числом j . Покажемо, що $g(j)$ — таке найбільше ціле число $i < j$, що $x_1x_2\dots x_i = x_{j-i+1}x_{j-i+2}\dots x_j$. Якщо такого i немає, то $g(j) = 0$.

За означенням функції відмов $g(1) = 0$. Нехай припущення індукції має місце для всіх $k < j$. При обчисленні $g(j)$ алгоритм ВІДМОВА, виконуючи оператор 4, порівнює x_j і $x_{g(j-1)+1}$.

Випадок 1. Нехай $x_j = x_{g(j-1)+1}$. Оскільки $g(j-1)$ — це таке найбільше i , що $x_1\dots x_i = x_{j-i}\dots x_{j-1}$, то рівність $g(j) = i + 1$ виконується. Таким чином, оператори 5 і 6 функцію $g(j)$ обчислюють правильно.

Випадок 2. Нехай $x_j \neq x_{g(j-1)+1}$. Тоді необхідно знайти найбільше значення i , для якого $x_1\dots x_i = x_{j-i}\dots x_{j-1}$ і $x_{i+1} = x_j$, якщо таке

число існує. Якщо такого i немає, то очевидно, що $g(j) = 0$, і $g(j)$ правильно обчислюється оператором 5. Нехай i_1, i_2, \dots — найбільше, друге за величиною і т. д. значення i , для яких $x_1 \dots x_i = x_{j-i} \dots x_{j-1}$. За допомогою індукції переконуємося, що $i_1 = g(j - 1)$, $i_2 = g(i_1)$, ..., $i_k = g(i_{k-1}) = g(j - 1)$, оскільки i_{k-1} — це $(k - 1)$ — е за величиною значення i , для якого

$$x_1 \dots x_i = x_{j-i} \dots x_{j-1},$$

а i_k — найбільше значення $i < i_{k-1}$, для якого

$$x_1 \dots x_i = x_{i_{k-1}-i+1} \dots x_{i_{k-1}} = x_{j-i} \dots x_{j-1}.$$

Оператор 4 розглядає по черзі i_1, i_2, \dots до тих пір, поки не знайде таке i , що $x_1 \dots x_i = x_{j-i} \dots x_{j-1}$ і $x_{i+1} = x_j$, якщо таке i існує. Після закінчення виконання оператора **Поки** $i = i_m$, якщо таке i_m існує, і, значить, $g(j)$ правильно обчислюється оператором 5.

Теорема доведена.

Теорема 10.2.2. Алгоритм ВІДМОВА обчислює значення функції g за час $O(l(p))$.

Доведення. Оператори 1 і 3 мають складність константи. Число виконань оператора **Поки** пропорційне числу зменшення значення i оператором 3 ($i := g(i - 1)$). Оскільки спочатку $i = 0$, а оператор 6 виконується не більше $l(p) - 1$ раз, то оператор **Поки** виконується не більше $l(p)$ разів. Отже, загальна складність алгоритму пропорційна величині $O(l(p))$.

Теорема доведена.

Алгоритм задання автомата A , про який ішла мова вище, тепер можна побудувати, використовуючи значення функції g . Спочатку будеться так званий скелетний автомат з $l(p) + 1$ станом, для якого функція переходів f визначається так: автомат переходить із стану $i - 1$ в стан i під дією лише символу x_i слова p (див. приклад 10.2.2), а потім функція переходів скелетного автомата довизначається за допомогою функції відмов. Цей алгоритм має такий вигляд.

АВТОМАТ (p)

/* змінна j — це номер символу в слові p */
початок

1. Для $j = 1$ до $l(p)$ виконати

$$f(j - 1, x_j) := j;$$
 кіц;
2. Для всіх x із X таких що $x \neq x_1$ виконати

$$f(0, x) := 0;$$
 кіц;
3. Для $j = 1$ до $l(p)$ виконати

Для всіх x із X таких що $x \neq x_{j+1}$ виконати
 $f(j, x) := f(g(j), x);$
 кц;
 кц;
кінець

Теорема 10.2.3. Алгоритм АВТОМАТ будеє автомат A , такий, що

$$f(a_0, b_1b_2\dots b_k) = j$$

тоді і тільки тоді, коли $x_1\dots x_j$ — суфікс слова $b_1b_2\dots b_k$ і $x_1\dots x_j$ не є суфіксом слова $b_1b_2\dots b_k$ при $i > j$.

Доведення цієї теореми майже дослівно повторює доведення теореми 10.2.1.

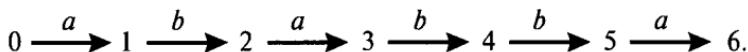
Зауважимо, що для того щоб перевірити, чи є слово q підсловом слова p , можна і не будувати повністю автомат. Для цього достатньо лише побудувати функцію відмов. Оскільки за теоремою 10.2.2 функція відмов обчислюється за час $O(l(q))$, то визнати входження q в p можна за час $O(l(q) + l(p))$.

Задача розпізнавання того, чи входять слова q_1, q_2, \dots, q_k в слово p , зводиться до попередньої задачі. Для її розв'язання спочатку будується скелетний автомат для слів q_1, q_2, \dots, q_k . Граф переходів скелетного автомата буде, очевидно, деревом. Для цього дерева знайдемо функцію відмов (це можна зробити за час $O(l) = O(l(q_1) + l(q_2) + \dots + l(q_k))$). Після цього описаним вище способом довизначаємо автомат, який необхідно було побудувати. Тоді за час $O(l + l(p))$ можна визначити, чи є слово q_i підсловом слова p .

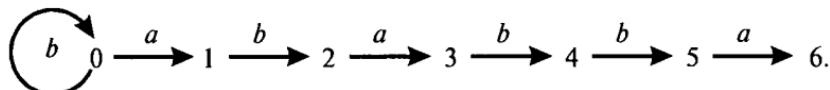
Приклад 10.2.2

Побудуємо автомат A для слова $p = ababba$, яке розглядалося в попередньому прикладі 10.2.1.

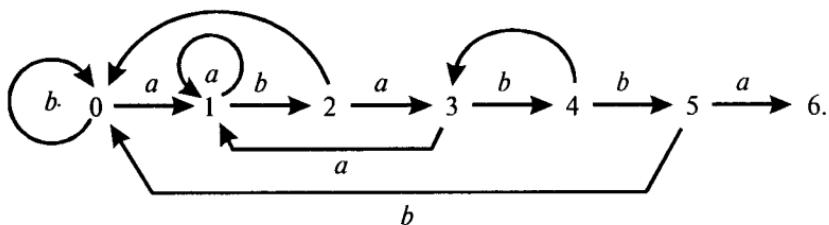
Після виконання первого циклу (оператор 1) в алгоритмі АВТОМАТ одержуємо такий граф переходів (скелетний автомат) шуканого автомата:



Після виконання другого циклу (оператор 2) алгоритму АВТОМАТ одержуємо такий граф переходів шуканого автомата:



Після виконання третього циклу (оператора 3) алгоритму АВТОМАТ одержуємо остаточний вид графа переходів автомата A :



Отже, автомат, який будується алгоритмом АВТОМАТ за словом p , представляє мову, що має вигляд $\{X\} \cdot p$. Користуючись даним методом розв'язання задачі ідентифікації слів, можна виконувати пошук і заміну відразу кількох слів, тобто побудувати автомат, який представляє мову $\{X\} \cdot (p_1 \vee p_2 \vee \dots \vee p_k)$.

Деякі системи підготовки та редакції текстів використовують так звані spell-checking підсистеми. Опишемо коротко принцип побудови та роботу такої підсистеми. Якщо довільна скінчена множина слів S вхідної напівгрупи $F(X)$ представляється скінченим автомatem, то цей факт можна застосувати для перевірки правопису слів. Дійсно, нехай множина S відповідає множині слів словника (наприклад, англійської мови). Тоді, якщо є сумнів відносно правильності написання деякого слова p із S , то достатньо слово p подати на вход автомата A , який представляє множину слів S деякою множиною станів F . Якщо автомат A під дією слова p досягає одного із станів множини F , то слово p написане правильно, а якщо ні, то в слові є помилка. Зауважимо, що така перевірка слова p вимагає часу, пропорційного довжині $l(p)$ слова p . Розглянемо на прикладі роботу такого автомата.

Приклад 10.2.3

Нехай $S = \{abase, abash, abat, abbey\}$ — підмножина слів англійської мови (словник). Відповідний автомат, який представляє слова із S , показаний на рис. 10.2.1. В цьому автоматі 0 — початковий стан, а 6, 7, 10 — заключні стани.

Для того, щоб застосувати автомат A для перевірки правопису слів із множини S та ідентифікації помилок у словах, введемо ще один стан — стан 11 — і переходи (показані пунктирами) в цей стан з кожного іншого стану, за винятком станів із F , під дією букв англійського алфавіту, для яких немає переходів в A .

Тепер для перевірки правильності написання конкретного слова, наприклад $abbei$, подаємо його на вход автомата A . Під

дією цього слова попадаємо в стан 11, який не належить множині заключних станів F , отже, слово *abbei* написане неправильнно. Переход у стан 11 був виконаний зі стану 9 під дією входного символу i , тому цей символ ідентифікується як помилковий. ▲

Зауважимо, що задання словника за допомогою скінченного автомата дає значну економію пам'яті ЕОМ, що позитивно позначається на всьому процесі роботи з таким словником. Як зазначає автор праці [82], словник, що включає 300 тис. слів англійської мови, вимагає 5,2 Мбайтів пам'яті в текстовому форматі, в той час як відповідний скінчений пам'яті.

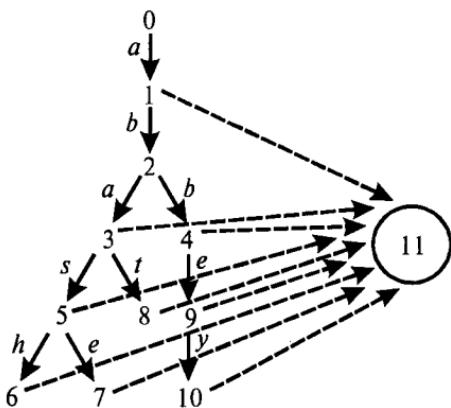


Рис 10.2.1.
Автомат A , який представляє слова із S
автомат вимагає лише 0,3 Мбайта

2. ПОНЯТТЯ ГІПЕРТЕКСТУ

Системи підготовки та редакції текстів знайшли подальший розвиток в системах роботи з гіпертекстами.

Гіпертекстом називається текст, в якому задані зв'язки між виділеними в ньому елементами. Як об'єкт, гіпертекст можна зобразити у вигляді графа, вершинам якого є деякі самостійні фрагменти інформації (тексти, формули, графічна, звукова інформація тощо), а ребрам — зв'язки між цими фрагментами. Отже, на відміну від баз даних елементи гіпертексту можуть бути неоднорідними об'єктами, що не потребують форматизації. Доступ до окремих фрагментів гіпертексту виконують системи навігації по гіпертексту, за допомогою яких проводиться обхід вершин графа гіпертексту. При цьому системи роботи з гіпертекстами дають у розпорядження користувача такої системи дружній інтерфейс.

Деталі технології роботи з гіпертекстами можна знайти в [23].

Контрольні питання

1. Який автомат називається універсальним програмним автоматом?
2. Що означає універсальність програмного автомата? Перелічіть функціонально повний набір операцій, які повинен виконувати автомат для забезпечення цієї повноти.
3. Дайте означення суфікса, підслова, суфікса слова.
4. Дайте означення функції відмов.
5. Який теоретичний факт покладено в основу алгоритму ідентифікації слів?
6. Що називається гіпертекстом?

Задачі і вправи

1. Побудуйте spell-checking підсистему для словника $S = \{\text{ідентифікація, ідентифікатор, ідентичний, ідентифікувати, ідеал}\}$.
2. Побудуйте функцію відмов для слова $p = aabbaab$.
3. Побудуйте автомат, який виконує ідентифікацію слова p із попередньої вправи 2.
4. Побудуйте автомат, який виконує ідентифікацію слів із множини S , даної у вправі 1.

§ 10.3. КОМП'ЮТЕРНА АЛГЕБРА

Комп'ютерна алгебра — це досить розвинена галузь знань, яка має свої власні методи досліджень [6, 26, 76]. Зрозуміло, що детальне викладення цих методів у такому короткому підрозділі неможливе, тому обмежимося лише розглядом деяких задач комп'ютерної алгебри.

1. ЗАДАЧА СПРОЩЕННЯ АЛГЕБРАЇЧНИХ ВИРАЗІВ

Однією з основних задач комп'ютерної алгебри є задача спрощення алгебраїчних виразів і рівнянь.

Задача спрощення алгебраїчних виразів і рівнянь має два різновиди:

- побудова за даним об'єктом простішого об'єкта, еквівалентного даному;
- знаходження єдиного представлення для еквівалентних об'єктів.

Більше того, нехай T — деяка множина алгебраїчних виразів, наприклад скінченних термів, у мові першого порядку, скін-

ченних класів логічних формул або скінченних класів програм, і нехай символ \sim означає відношення еквівалентності на множині T .

Задача побудови еквівалентного заданому простішого об'єкта полягає в тому, щоб знайти ефективну процедуру $S: T \rightarrow T$ таку, що для довільного t із T справедливі властивості

$$S(t) \sim t, \quad (SE)$$

$$S(t) \leq t, \quad (SS)$$

де \leq — деяке відношення спрощення, яке може означати, наприклад, що для деякого $s = S(t)$ нерівність " $s \leq t$ " справедлива тоді і тільки тоді, коли " s не довший за t " або "для представлення s в ЕОМ потрібно не більше пам'яті, ніж для t ". Звичайно, s коротше, ніж t , якщо $S(t) < t$.

Задача обчислення єдиного представлення для еквівалентних об'єктів (задача канонічного спрощення) полягає в тому, щоб знайти ефективну процедуру $S: T \rightarrow T$ для відношення \sim , таку, що для довільних s, t із T справедливі такі властивості:

$$S(t) \sim t, \quad (SE)$$

$$s \sim t \Rightarrow S(s) = S(t), \quad (SC)$$

тобто S вказує на єдиного представника в кожному класі еквівалентності, і цей елемент $S(t)$ називається *канонічною формою* об'єкта t .

Для формулювання наведеної нижче твердження, яке характеризує поняття канонічної форми, нагадаємо, що підмножина A множини натуральних чисел N називається *рекурсивною*, якщо її характеристична функція є обчисленою (яку можна обчислити) функцією.

Нехай T — деяка множина алгебраїчних виразів і $h: T \rightarrow N$, де N — множина натуральних чисел. Множина виразів T називається *розв'язною*, якщо множина значень функції h рекурсивна, тобто множина номерів виразів із S має обчисленну характеристичну функцію.

Теорема 10.3.1 (канонічне спрощення і розпізнавання еквівалентності). Нехай T — множина об'єктів, алгоритмічно розв'язна, і символ \sim означає відношення еквівалентності на T . Відношення \sim алгоритмічно розв'язне на T тоді і тільки тоді, коли існує канонізація S для відношення \sim .

Д о в е д е н н я. З одного боку, справедливість цієї теореми випливає з властивостей (SE) і (SC) . Дійсно, згідно з властивостями (SE) і (SC) маємо $s \sim t \Leftrightarrow S(s) = S(t)$.

З іншого боку, нехай g — обчислена функція, яка відображає N на T . Покладемо $S(s) = g(n)$, де n — найменше натуральне число, для якого $g(n) \sim s$. Функція S обчислена, бо g обчислена. Справедливість властивостей (SE) і (SC) очевидна.

Теорема 10.3.2 (канонічне спрощення і обчисленність). *Нехай T — розв’язна множина виразів, ω — обчислена бінарна операція на T , і символ \sim означає деяке відношення еквівалентності, яке є відношенням конгруентності відносно ω . Припустимо, що задана канонізація S для відношення \sim . Введемо множину “канонічних представників” (достатня множина) $Rep(T) = \{t \in T \mid S(t) = t\}$ і визначимо $\omega'(s, t) = S(\omega(s, t))$ для всіх $s, t \in Rep(T)$. Тоді алгебра $(Rep(T), \Omega')$ ізоморфна фактор-алгебрі $(T/\sim, \Omega)$ і ω' обчислена (тут операція $\omega(C_s, C_t) = C_{\omega(s, t)}$, де C_t — клас еквівалентності відносно \sim , який містить t).*

Ця теорема показує, що при наявності канонізації для відношення еквівалентності, яке є відношенням конгруентності відносно деякої обчисленої операції, повинен існувати алгоритм для виконання операції у фактор-алгебрі. Але виявляється, що канонізація для відношення еквівалентності на заданій множині виразів не завжди можлива. Ця обставина є теоретичною передумовою розгляду неканонічного спрощення. Для деяких досить простих класів виразів задача функціональної еквівалентності алгоритмічно не розв’язується, тобто для них не існує якоїсь канонізації. Двома найпоширенішими є спрощення типу 0-еквівалентність (або нормальнє) і регулярне спрощення.

Спрощення типу 0-еквівалентність може бути визначене у випадку, коли дана множина виразів містить елемент, який відіграє роль нульового. Обчисленне відображення $S: T \rightarrow T$ називається *відображенням типу 0-еквівалентності для відношення \sim на множині T* , якщо для всіх елементів t із T виконано

$$S(t) \sim t, \quad (SE) \quad t \sim 0 \Rightarrow S(t) = S(0). \quad (SZ)$$

Очевидно канонізація є нормальним спрощенням. З іншого боку, якщо існує така обчислена операція ω на T , що для довільного $s, t \in T$ справедливе відношення

$$s \sim t \Leftrightarrow \omega(s, t) \sim 0, \quad (ME)$$

то наявність нормального спрощення веде за собою існування канонізації для відношення \sim .

Поняття регулярного спрощення використовується в ситуаціях, коли серед виразів є трансцендентні функції (наприклад, \exp , \log тощо). Регулярне спрощення гарантує, що всі нерациональні терми в спрощених виразах алгебраїчно незалежні.

2. ВАЖЛИВІ СПРОЩЕННЯ

Спрощення алгебраїчних виразів означає спрощення логічних формул і “оптимізацію” програм. Обидві задачі важливі для автоматичного проектування і контролю математичного забезпечення ЕОМ.

Важливу практичну роль у системах комп’ютерної алгебри відіграють процедури спрощення алгебраїчних виразів, призначені не стільки для канонізації, скільки для фактичного “спрощення” виразів відносно різноманітних інтуїтивних критеріїв простоти (іншими словами, неканонічне спрощення).

Спрощення термів. Нехай $T(\Omega, X)$ — Ω -алгебра термів над алфавітом X сигнатури Ω .

Гомоморфізм (ендоморфізм) $\sigma: T(\Omega, X) \rightarrow T(\Omega, X)$, який задовольняє умову $\sigma(x) = x$ для всіх $x \in X$, крім скінченного їх числа, називається *підстановкою*.

Присвоюванням (інтерпретацією) називається гомоморфізм $V: T(\Omega, X) \rightarrow A$, де A — деяка фіксована Ω -алгебра. $V(t)$ називається *значенням терма t* відносно присвоювання V . Терми t і t' називаються *функціонально еквівалентними*, якщо $V(t) = V(t')$.

Нехай $E \subseteq T(\Omega, X) \times T(\Omega, X)$. Пара $(a, b) \in E$ буде представлена, як ліва і права частини рівняння. Наступні два відношення еквівалентності на $T(\Omega, X)$, пов’язані з множиною E , є основними для всіх алгебраїчних теорій:

$E \models s = t$ ($s = t$ семантично рівні в теорії E)

$E \mid -s = t$ ($s = t$ доводиться як наслідок в теорії E).

Означення 10.3.1 ($E \models s = t$). Якщо A — Ω -алгебра, то формула $s = t$ істинна в A ($E \models s = t$) тоді і тільки тоді, коли $s = t$ істинна в усіх Ω -алгебрах, які визначаються множиною E .

Означення 10.3.2 ($E \mid -s = t$). Відношення $E \mid -s = t$ істинне, якщо формула $s = t$ може бути виведена за скінченне число кроків у такому численні (числення рівностей):

1) $a = b$ для довільних $(a, b) \in E$ (елементи E — це аксіоми);

2) $a = a$ (рефлексивність),

$a = b \Rightarrow b = a$ (симетричність),

$a = b, b = c \Rightarrow a = c$ (транзитивність);

3) $a = b \Rightarrow \sigma(a) = \sigma(b)$ для довільної підстановки σ (правило підстановки);

4) $a_1 = b_1, \dots, a_n = b_n \Rightarrow \omega(a_1, \dots, a_n) = \omega(b_1, \dots, b_n)$ для довільної операції $\omega \in \Omega$ арності n ($n = 1, 2, \dots$) (правило заміни).

Числення рівностей є повним і правильним відносно означеного вище відношення \models у зв’язку з такою теоремою.

Теорема Біркгофа. $E \models s = t \Leftrightarrow E \mid s = t$.

За цією теоремою відношення семантичної рівності є рекурсивно переліченним, якщо E — рекурсивно переліченна множина [6]. Далі для фіксованого E відношення $=_E$ означає таке відношення еквівалентності: $s =_E t \Leftrightarrow E | - s = t$.

3. ПОВНІ СИСТЕМИ РЕДУКЦІЙ, КРИТИЧНІ ПАРИ, АЛГОРИТМ ПОПОВНЕННЯ

В класах T алгебраїчних об'єктів із відношенням еквівалентності \sim часто можна природним шляхом визначити *відношення редукції* \rightarrow , яке описує, як “можна перейти за один крок від об'єкта s до простішого еквівалентного об'єкта t ”.

Ітерація цієї редукції являє собою канонізацію для відношення \sim , якщо виконані такі умови:

- відношення \rightarrow може бути реалізоване алгоритмічно і, крім того, рефлексивне, симетричне, транзитивне замикання відношення редукції \rightarrow збігається з відношенням \sim ;
- відношення \rightarrow повне, тобто:

а) ітерація процесу редукції, яка починається з одного і того ж об'єкта, завжди завершується отриманням одного і того ж об'єкта, що не редукується далі (*единість*).

б) ітерація процесу редукції завжди завершується після скінченного числа кроків об'єктом, що далі не редукується (*завершеність*).

Для конкретного відношення редукції \rightarrow доведення властивості завершеності інколи потребує спеціальної техніки.

Перевірка властивості єдиності може бути алгоритмізована з використанням двох таких важливих ідей:

- локалізація (використовується в загальному випадку);
- метод критичних пар (що використовуються для відношень редукції, “які породжуються” скінченним числом схем редукцій).

Скінченнопороджені відношення редукції, що не є повними, можуть бути поповнені шляхом добавлення схем редукції, які виникають при аналізі критичних пар. Цей метод *поповнення* разом із локалізацією і методом аналізу критичних пар становить третю основну ідею. Ці ідеї мають широке застосування методологічного характеру для побудови канонізацій.

Більш формально, нехай $T \neq \emptyset$ — довільна множина алгебраїчних виразів. У контексті даного розділу довільне бінарне відношення $\rightarrow \subseteq T \times T$ буде називатися *відношенням редукції*. Обернене відношення \leftarrow , транзитивне замикання \rightarrow^+ , рефлексивно-транзитивне замикання \rightarrow^* і, нарешті, рефлексивно-транзи-

тивно-симетричне замикання \leftrightarrow^* відношення \rightarrow мають наведені позначення. Відношення \rightarrow^0 означає відношення тотожності і $\rightarrow^{n+1} = \rightarrow \cdot \rightarrow^n$.

Означення 10.3.3. 1. Відношення \rightarrow називається *ньотеровим*, якщо воно задовільняє властивість завершенності.

2. Якщо відношення \rightarrow зрозуміле з контексту, то через x позначається нормальна форма елемента x із T відносно \rightarrow , тобто не існує такого $y \in T$, що $x \rightarrow y$.

3. Елемент $z \in T$ називається спільним нащадком елементів $x, y \in T$, якщо $x \rightarrow^* z \leftarrow^* y$ (це позначається $x \downarrow y$).

Нехай символ \sim означає відношення еквівалентності на T , а символ \rightarrow означає деяку ньотерову редукцію, таку, що $\leftrightarrow^* = \sim$. Припустимо, що задана така функція "вибору" $\text{Sel}: T \rightarrow T$, що $x \rightarrow \text{Sel}(x)$ для довільного $x \in T$, який не знаходиться в нормальній формі. Розглянемо обчисленну функцію S , означення якої дається рекурсивно:

$$S(x) = \begin{cases} x, & \text{якщо } x \text{ в нормальній формі,} \\ S(\text{Sel}(x)) & \text{інакше.} \end{cases}$$

Наземо S такого типу алгоритмом нормальній форми для редукції \rightarrow . Наша мета — знайти алгоритмічний тест для визначення того, чи є S канонізацією. Зауважимо, що S задовільняє аксіому (SE) , тобто $S(x) \leftrightarrow^* x$. До того ж справедливе твердження для довільного $x \in T$ ($x \rightarrow^* S(x)$) (SE') .

Нижче доведено ряд лем, які показують, що тест для (SC) , тобто для властивості $x \leftrightarrow^* y \Rightarrow S(x) = S(y)$, може бути приведений інтуїтивно до простіших тестів і, врешті-решт, може бути "локалізований". А метод критичних пар дозволяє зробити тест ефективним.

Нехай \rightarrow — ньотерове відношення редукції і S — алгоритм приведення до нормальній форми для \rightarrow . Справедливі такі твердження.

Лема 10.3.1 (зведення канонічності до умови Черча–Россера). Алгоритм S є канонізацією для відношення \leftrightarrow^* в тому і тільки в тому випадку, коли \rightarrow має властивість Черча–Россера, тобто для довільних $x, y \in T$ із $x \leftrightarrow^* y$ маємо $x \downarrow y$.

Д о в е д е н н я. Із властивості канонізації відношення \leftrightarrow^* очевидним чином випливає властивість Черча–Россера. Навпаки, нехай $x \leftrightarrow^* y$, тоді для деякого z маємо $S(x) \leftarrow^* x \rightarrow^* z \leftarrow^* y \rightarrow^* S(y)$ внаслідок властивості Черча–Россера. Повторно застосовуючи властивість Черча–Россера до $S(x)$ і z , $S(y)$ і z , одержуємо рівності $S(x) = z$ і $z = S(y)$, що й потрібно було довести.

Лема 10.3.2 (зведення умови Черча–Россера до умов злиття).
 Відношення \rightarrow має властивість Черча–Россера тоді і тільки тоді, коли воно задовольняє умову злиття, тобто для довільних $x, y, z \in T$, якщо $x \leftarrow z \rightarrow^* y$, то $x \downarrow y$.

Д о в е д е н н я. Із властивості Черча–Россера очевидним чином випливає справедливість умови злиття. Навпаки, оскільки відношення \leftrightarrow^* являє собою об'єднання всіх \leftrightarrow^n , то можна скористатися індукцією за числом n . Базис індукції для $n = 0$ виконується очевидним чином. Якщо $x \leftrightarrow^{n+1} y$, то або $x \rightarrow z \leftrightarrow^n y$, або $x \leftarrow z \leftrightarrow^n y$ для певного z . В обох випадках із припущення індукції випливає справедливість $z \rightarrow^* u \leftarrow^* y$ для деякого u . У першому випадку маємо $x \rightarrow^* u \leftarrow^* y$ і, отже, $x \downarrow y$, у другому випадку на основі властивості злиття маємо $x \rightarrow^* v \leftarrow^* u$ і для деякого v , тому $x \rightarrow^* v \leftarrow^* u \leftarrow^* y$, тобто знову $x \downarrow y$.

Лема 10.3.3 (зведення умови злиття до умови локального злиття). Відношення \rightarrow задовольняє умову злиття тоді і тільки тоді, коли \rightarrow задовольняє умову локального злиття, тобто для довільних $x, y, z \in T$, якщо $x \leftarrow z \rightarrow y$, то $x \downarrow y$.

Д о в е д е н н я. Із другої умови, тобто умови локального злиття, очевидним чином випливає перша умова. Навпаки, нехай $z_0 \in T$ – довільний фіксований елемент. Користуючись методом трансфінітної індукції відносно відношення порядку \rightarrow^* , доведемо таке твердження: для всіх z , таких, що $z_0 \rightarrow^* z$, і для довільних x, y , якщо $x \leftarrow^* z \rightarrow^* y$, то $x \downarrow y$. Покажемо, що для всіх x, y , якщо $x \leftarrow^* z \rightarrow^* y$, то $x \downarrow y$. Випадки, коли $x = z_0$ і $z_0 = y$, очевидні. У протилежному випадку $x \leftarrow^* x_1 \leftarrow z_0 \rightarrow y_1 \rightarrow^* y$ для деяких x_1, y_1 . Тоді згідно з умовою локального злиття і припущенням індукції існують елементи u, v, w , які задовольняють такі умови:

$$x_1 \leftarrow^* z_0 \rightarrow^* y_1, \quad x \leftarrow^* x_1 \rightarrow^* u, \\ u \leftarrow^* y_1 \rightarrow^* y, \quad v \rightarrow^* w, \quad y \rightarrow^* w,$$

що і потрібно було показати.

З доведених вище лем випливає такий критерій для перевірки канонічності відображення S .

Лема 10.3.4 (локальний критерій канонічності). Відображення S є канонізацією для відношення \leftrightarrow^* тоді і тільки тоді, коли для довільних x, y, z із $x \leftarrow z \rightarrow y$ випливає $S(x) = S(y)$.

Цей критерій все ще не є ефективним, оскільки для його перевірки в загальному випадку необхідно розглянути нескінченно багато x, y, z . Однак цей критерій стає ефективним у випадку скінченного числа схем редукцій.

4. АЛГОРИТМ КНУТА—БЕНДІКСА КРИТИЧНОЇ ПАРИ

Алгоритм Кнута—Бендікса критичної пари використовується для знаходження розв'язку задачі спрощення для відношень еквівалентності \equiv_E на множинах термів, де E — система рівностей алгебри $T(\Omega, X)$.

Будемо зображати терми з $T(\Omega, X)$ у вигляді дерев. Наприклад, $t = g f x y g x y 3 x$ має зображення у вигляді дерева (рис. 10.3.1). Будемо говорити, що підтерм $f x y$ входить у t на місці (1), підтерм $g x y 3$ входить у t на місці (2), підтерм y входить у t на місці (1, 2) і т. д. Позначимо це так: $t/(1) = f x y$, $t/(2) = g x y 3$, $t/(1, 2) = y$ і т. д. $t/e = t$, де e — пуста послідовність місць.

Множиною всіх місць $Oc(t)$ терма t в даному прикладі є множина $\{e, (1), (2), (3), (1, 1), (1, 2), (2, 1), (2, 2), (2, 3)\}$.

Позначимо символом \cdot конкатенацію на множині $F(N)$ всіх місць. *Префіксне упорядкування* на $F(N)$ визначається таким чином ($F(N)$ — вільна напівгрупа слів над алфавітом $N = \{1, 2, \dots\}$): $u \leq v \Leftrightarrow u \cdot w = v$, де w — відповідне слово. Покладемо $u/v = w$, якщо $u \cdot w = v$, і $u \setminus v$ (u і v називаються *диз'юнктними*), якщо не виконується ні $u \leq v$, ні $v \leq u$.

Нарешті, для $s, t \in T(\Omega, X)$ і місця u позначимо $t[u \leftarrow s]$ терм, який отримано з t заміною підтерма, який стоїть на місці u , термом s . Наприклад, у наведеному вище прикладі $t[2 \leftarrow f f x x 2] = g f x y g x y 3 x$.

Нехай E — система рівностей з $T(\Omega, X)$. Відношення редукції \rightarrow_E на $T(\Omega, X)$ можна визначити звичайним чином так, що $s \rightarrow_E t$, якщо t одержується з s застосуванням однієї з рівностей із E як “правила перетворення” (зліва направо).

Означення 10.3.4. Говорять, що s редукується до t за допомогою E (це позначається $s \rightarrow_E t$), якщо знайдеться таке правило $(a, b) \in E$, а також підстановка σ і таке місце $u \in Oc(s)$, що $s/u = \sigma(a)$ і $t = f[u \leftarrow \sigma(b)]$.

Приклад 10.3.1

Нехай E — система аксіом для груп, тобто $E = \{(1) 1 \cdot x = x, (2) x \cdot x^{-1} = 1, (3) (xy)z = x(yz)\}$. Тоді $(xx^{-1})z \rightarrow_E 1 \cdot z \rightarrow_E z$.

Надалі допускаються тільки такі системи E , які мають властивість:

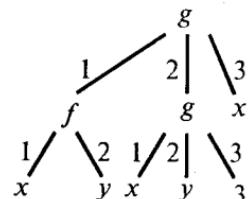


Рис. 10.3.1.
Дерево терма t

- для довільних $(a, b) \in E$ $\text{var}(a) \supseteq \text{var}(b)$, де $\text{var}(t)$ — множина змінних терма t .

Означення 10.3.5. Говорять, що терми p і q утворюють критичну пару в E , якщо знайдуться такі правила $(a_1, b_1), (a_2, b_2) \in E$ і таке місце $u \in Oc(a_1)$, що:

- a_1/u не є змінною;
- a_1/u та a_2 терми, що уніфікуються;
- $p = \sigma(a_1) [u \leftarrow \sigma_2(b_2)]$ та $q = \sigma_1(b_1)$, де σ_1, σ_2 — підстановки, причому $\sigma_1(a_1/u) = \sigma_2(a_2)$, які являють собою НЗУ термів a_1/u та a_2 , тобто p та q одержуються шляхом застосування правил (a_1, b_1) та (a_2, b_2) до НЗУ(a_1, a_2).

Приклад 10.3.2

Якщо $T(\Omega, X)$ являє собою вільну групу, то покладемо $a_1 = (xy)z, b_1 = x(yz), a_2 = xx^{-1}, b_2 = 1, u = (1)$. Тоді xx^{-1} є найзагальніший уніфікатор термів $a_1/u = xy$ і $a_2 = xx^{-1}$ (так що виконується умова для змінних, візьмемо $\sigma_1(y) = x$ та $\sigma_1(y) = x^{-1}, \sigma_2$ — тотожна підстановка). Отже, терми $p = \sigma_1(a_1) [u \leftarrow \sigma_2(b_2)] = (x^{-1}x)z [(1) \leftarrow 1] = 1 * z$ і $q = \sigma_1(b_1) = x^{-1}(xz)$ утворюють критичну пару в E . ▲

Теорема Кнута–Бендікса (зведення умови локального злиття до умови злиття критичних пар). *Відношення редукції \rightarrow_E задоволяє умови локального злиття, якщо для довільної критичної пари (p, q) в E справедливе $p \downarrow_E q$ [26].*

Якщо E — скінчена множина, то завжди можна побудувати алгоритм Sel, такий, що $t \rightarrow_E Sel(t)$, коли t не в нормальній формі. Якщо \rightarrow_E ньютерове, то позначимо S_E алгоритм нормальної форми, який базується на алгоритмі Sel.

Алгоритм критичної пари для правил перетворення

Вхід: E (система рівностей).

Питання: Чи буде алгоритм S_E нормальної форми канонізацією для відношення $=_E$?

1. Покласти C рівним множині критичних пар з E .

2. Для всіх $(p, q) \in C$ виконати

$$(p_0, q_0) = S_E(p, S_E(q)).$$

Якщо $p_0 \neq q_0$ то відповідь “ S_E не канонічний”.

Кінчили.

Відповідь: “ S_E канонічний”.

Коректність цього алгоритму є простим наслідком наведених вище теорем та лем.

Означення 10.3.6. Відношення \rightarrow_E називається *стійким*, якщо для довільних термів s, t і підстановки σ має місце $s \rightarrow_E t \Rightarrow \sigma(s) \rightarrow_E \sigma(t)$, і *сумісним*, якщо для довільних s, t_1, t_2 та $u \in Oc(s)$ спрavedливе $t_1 \rightarrow_E t_2 \Rightarrow s[u \leftarrow t_1] \rightarrow_E s[u \leftarrow t_2]$.

Приклад 10.3.3

У наведеному вище прикладі 10.3.2 з теорії груп критична пара (p, q) , що виникає з правил (3) і (2) множини E , не має загального уніфікатора. Єдина можлива для p редукція — це $p = 1 * z = z$ і $q = x^{-1}(xz)$ — вже знаходиться в нормальній формі. Через це відношення \rightarrow_E не задовольняє умову локального злиття, і алгоритм S_E нормальної форми не є канонізацією.

Наведена нижче система аксіом E' для теорії груп є ньютеровою і знаходиться за системою E шляхом її поповнення деякими тотожностями:

- | | | |
|-----------------------------------|------------------------|------------------------|
| (1) $1 \cdot x = x$, | (2) $x^{-1}x = 1$, | (3) $(xy)z = x(yz)$, |
| (4) $1^{-1} = 1$, | (5) $x^{-1}(xy) = y$, | (6) $x \cdot 1 = x$, |
| (7) $(x^{-1})^{-1} = x$, | (8) $xx^{-1} = 1$, | (9) $x(x^{-1}y) = y$, |
| (10) $(xy)^{-1} = y^{-1}x^{-1}$. | | |

За допомогою скінченного числа перевірок можна встановити справедливість таких рівностей $S_{E'}(p) = S_{E'}(q)$ для всіх критичних пар із E' . Зокрема, наведена вище критична пара має загальний образ z . Таким чином, за теоремою Кнута–Бендікса відношення \rightarrow_E задовольняє умову (локального) злиття, та S_E' представляє канонізацію для відношення $\leftrightarrow_E' = \leftrightarrow_E = =_E$. Це означає, що перевірка тотожностей у теорії груп може бути повністю автоматизована. ⁴

5. АЛГОРИТМ ПОПОВНЕННЯ КНУТА–БЕНДІКСА

Якщо алгоритм Кнута–Бендікса критичної пари застосовний до системи рівностей E і показує, що редукція критичної пари (p, q) приводить до нерівності $S(p) \neq S(q)$, то слід спробувати додати до E правило $S(p) = S(q)$ або $S(q) = S(p)$ для досягнення повноти: дійсно, рефлексивно-симетрично-транзитивне замикання відношення \rightarrow_E не змінюється після такого додавання, тоді як саме відношення \rightarrow_E розширюється. Цей процес можна ітерувати доти, поки є надія, що буде досягнуте “насичення”, тобто критична пара буде мати єдину нормальну форму. В цьому випадку побудована таким чином система рівностей знову породжує те ж саме відношення еквівалентності $=_E$, що і початкова система, але відповідне їй відношення редукції \rightarrow_E єдине, тобто асоційований

алгоритм нормальної форми являє собою канонізацію для відношення $=_E$. Перед розширенням E необхідно перевірити, чи залишається відношення \rightarrow_E ньютеровим після розширення. Для того щоб забезпечити цю умову, потрібно вибрати одну з двох можливих пар: $(S(p) = S(q))$ або $(S(q) = S(p))$. Може статися, що збереження властивості завершення неможливо довести для жодної з цих пар. У цьому випадку процес поповнення не може бути розумно продовжений. Нижче дається приблизна схема цієї процедури.

Алгоритм (поповнення для правил перетворення)

Дано: E — скінчена система рівностей, така, що \leftrightarrow_E ньютерове.

Знайти: F — скінчену систему рівностей, таку, що “ \leftrightarrow_E^* ” = \leftrightarrow_F^* і “ \rightarrow_F ” задовольняє умову (локального) злиття (тобто S_F — канонізація).

- 1) $F = E$;
- 2) $C = \{ \text{множина критичних пар із } F \}$;
- 3) поки $C \neq \emptyset$ виконувати

взяти (p, q) із C .

$(p_0, q_0) = (S_F(p), S_F(q))$.

якщо $p_0 \neq q_0$ то

(Аналізувати (p_0, q_0)).

Добавити до C множину критичних пар $((p_0, q_0), F)$.

Добавити до F пару (p_0, q_0)

Вилучити з C пару (p, q) .

- 4) Зупинитися в разі успіху.

Підпрограми знаходження “множини критичних пар” і взяти елемент із трудношів не викликають. Підпрограма побудови “множини критичних пар” знаходить критичні пари, отримані з нового правила (p_0, q_0) та правил з F . Підпрограма Аналізувати визначає, чи залишається \rightarrow_E ньютеровим, коли (p_0, q_0) або (q_0, p_0) добавляються до E . У першому випадку пара (p_0, q_0) не змінюється, у другому випадку p_0 і q_0 міняються місцями. Якщо не виконується жодна з альтернатив, то підпрограма Аналізувати зупиняється з відповіддю “неуспіх”. При цьому можливі такі варіанти.

1. Алгоритм зупиняється з позитивною відповіддю. У цьому випадку отримана в кінці роботи алгоритму система F задовольняє вимоги, сформульовані на початку опису алгоритму.

2. Алгоритм зупиняється з відповіддю “неуспіх”. У цьому випадку не можна сказати нічого певного.

3. Алгоритм не зупиняється. У цьому випадку алгоритм являє собою процедуру переліку елементів відношення $=_E$. Можна сказати, що для відношення $s =_E t$ можливий перелік його елементів за допомогою редукцій s і t по відношенню до системи рівносостей, яка розширяється в процесі роботи алгоритму.

6. ПРОБЛЕМА ЗАГАЛЬНИХ ПІДВИРАЗІВ І КАНОНІЗАЦІЇ ДЛЯ ВІЛЬНИХ ГРУП

Розглянемо деякі конкретні відношення спрощення для алгебри термів (абсолютно вільна алгебра) і теорії вільних груп (побудова канонічної форми елементів вільної групи і проблема побудови базису скінченно породженої підгрупи вільної групи).

6.1. ЗАДАННЯ ТЕРМІВ ЗА ДОПОМОГОЮ СКІНЧЕННИХ АВТОМАТІВ

Нехай $T(\Omega, X)$ — абсолютно вільна алгебра термів і $t \in T(\Omega, X)$. Ми знаємо, що t можна задати у вигляді складеного об'єкта, який може бути деревом або ацикличним графом. Нехай $t = \omega_1(\omega_2(z_1, \omega_3(z_2, z_1, z)), \omega_1(z_1, \omega_2(z_1, z_2)))$ (рис. 10.3.2).

Позначимо:

$$x_1 = (\omega_1, 1), \quad x_2 = (\omega_1, 2), \quad x_3 = (\omega_2, 1), \quad x_4 = (\omega_2, 2), \quad x_5 = (\omega_3, 1), \\ x_6 = (\omega_3, 2), \quad x_7 = (\omega_3, 3), \quad x_8 = (z_1, 1), \quad x_9 = (z_2, 1), \quad x_{10} = (z, 1).$$

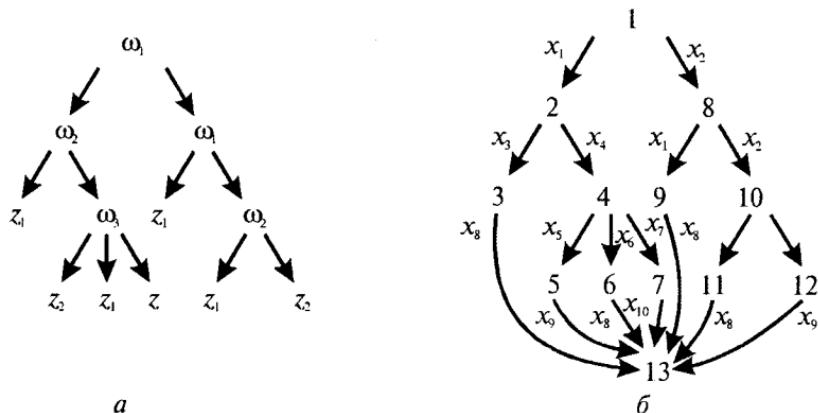


Рис. 10.3.2. Задання термів:
а — граф терма t ; б — автомат, який задає терм t

Розглянемо автомат
 $A = (\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13\},$
 $X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}, f, F = \{13\}),$
де f — функція, задана графом переходів.

Таким чином, щоб виконати переход від графа $G_t^h(v_0)$ до відповідного йому автомата, необхідно взяти число станів автомата $|V| + 1$, тобто поповнити множину V ще одним станом v' і визначити переходи в цей та інші стани автомата. Отже, автомат має на один стан більше, ніж граф, який зображає терм t , і не більше, ніж $2m$ переходів, де m — число дуг у цьому графі.

Для виконання переходу від графа $G_t^h(v_0)$, який зображає терм t за допомогою функції позначення h , де $v_0 \in V$ — початкова вершина цього графа, до відповідного йому автомата A занумеруємо елементи із X і Ω (ці множини скінченні в реальних системах комп'ютерної алгебри) цілими числами від 1 до n . Розіб'ємо вершини цього графа на класи за їх позначками, тобто v і v' належать одному класу еквівалентності, коли $h(v) = h(v')$. Отже, множина V розбивається на класи, які ми представимо масивом Отм довжини $|V|$, де $\text{Отм}(v)$ означає ім'я класу, який включає v . Іменем класу є номер цього класу в порядку появи класів. Кожному такому класу поставимо у відповідність число r — арність відповідної операції ω із Ω , символом якої позначені вершини даного класу, а класам, вершини яких позначені символами із X , припишемо арність 1. Нехай класам відповідають числа r_1, r_2, \dots, r_k .

Перенумеруємо вершини графа $G_t^h(v_0)$ і виконаємо обхід його вершин. Нехай u — вершина, що розглядається, лежить у класі, ім'я якого p , тобто $\text{Отм}(u) = p$, і наступною вершиною після вершини u при обході є вершина v , яка досягається по i -й дузі вершини u . Тоді в автоматі визначається переход із стану u в стан v під дією символу $x_i = (\omega, i)$, де

$$j = \begin{cases} i, & \text{якщо } \text{Отм}(u) = 1; \\ r_1 + \dots + r_i + i, & \text{якщо } \text{Отм}(u) = p > 1. \end{cases}$$

Якщо ж $r_p = 0$ для u , то із u визначається переход в v' під дією $x_j = (g(u), 1)$, де

$$j = \begin{cases} 1, & \text{якщо } \text{Отм}(u) = 1; \\ r_1 + \dots + r_i + 1, & \text{якщо } \text{Отм}(u) = p > 1. \end{cases}$$

При таких домовленостях побудова автомата A за відповідним графом виконується за один обхід графа, тобто вимагає часу, пропорційного числу дуг у графі.

Перш ніж перейти до розгляду задач і алгоритмів їх розв'язання, зробимо одне зауваження про задання термів скінченними автоматами.

Нехай t — деякий терм із $T(\Omega, X)$, $S(t)$ — ініціальний складений об'єкт, який представляє терм t , $G_t^h(v_0) = (V, E)$ — граф складеного $S(t)$, $h: V \rightarrow \Omega \cup X$ — функція позначення, і, нарешті, $A = (V \cup \{v^*\}, X, f, v_0, v^*)$ — скінчений ініціальний автомат, побудований за графом $G_t^h(v_0)$ так, як це було описано вище. Але такий спосіб кодування символів алфавіту X в даному випадку не найбільш економний. Враховуючи специфіку автомата A , можна запропонувати практичніший спосіб його задання. Ця специфіка полягає в тому, що після виконання мінімізації автомата A в один клас еквівалентності попадають ті і тільки ті стани автомата A , позначки яких у графі $G_t^h(v_0)$ збігаються. Ця обставина дозволяє ввести таке відношення еквівалентності R на множині вершин V графа $G_t^h(v_0)$: $uRv \Leftrightarrow h(u) = h(v)$. Тоді множина вершин V розбивається на класи еквівалентності V_1, V_2, \dots, V_k за цим відношенням, де $k < |\Omega \cup X|$ — число різних позначок вершин у графі $G_t^h(v_0)$. Оскільки граф $G_t^h(v_0)$ ізоморфно вкладається в граф переходів автомата A , то починати мінімізацію автомата A можна з розбиття $(V_1, \dots, V_k, \{v^*\})$, а не з розбиття $(V, \{v^*\})$. Це дає можливість елементами алфавіту X вважати порядкові номери дуг 1, 2, ..., q графа $G_t^h(v_0)$, де $q \leq \max\{\text{ar}(\omega) \mid \omega \in \Omega\}$, а ar — функція арності для операцій. Якщо в першому випадку $|X| \leq q|\Omega|$, то в другому $|X| \leq q$.

Приклад 10.3.4

Нехай $\Omega = \{+, -, \cdot, /\}, X = \{x, y, a, b\}$ і $t = x \cdot (a / b) + (x - y)$. Тоді в першому випадку маємо $X = \{x_1 = (+, 1, \cdot), x_2 = (+, 2, -),$

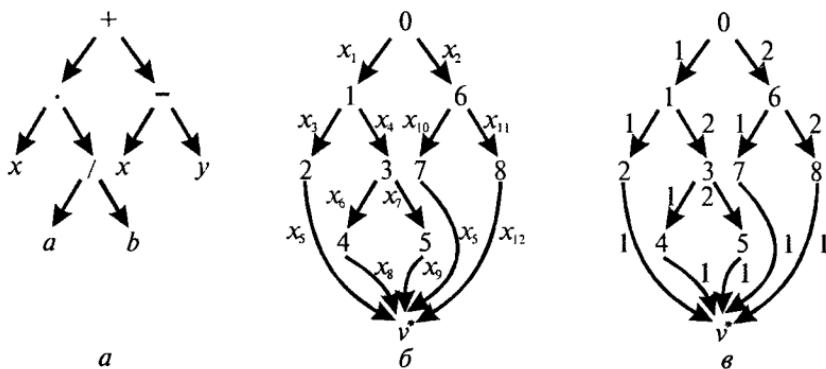


Рис. 10.3.3. Графи автоматів:

a — граф складеного $S(t)$; b — автомат A (перший спосіб);
 c — автомат A (другий спосіб)

$x_3 = (\cdot, 1, x)$, $x_4 = (\cdot, 2, /)$, $x_5 = (x, 1)$, $x_6 = (/, 1, a)$, $x_7 = (/, 2, b)$,
 $x_8 = (a, 1)$, $x_9 = (b, 1)$, $x_{10} = (-, 1, x)$, $x_{11} = (-, 2, y)$, $x_{12} = (y, 1)$ }, а
в другому $X = \{1, 2\}$ і одержуємо графи автоматів, зображені на
рис.10.3.3.

Таблиці переходів для першого і другого способів мають вигляд:

f	0	1	2	3	4	5	6	7	8	v^*
1	1	2	v^*	4	v^*	v^*	7	v^*	v^*	—
2	6	3	—	5	—	—	8	—	—	—

f	0	1	2	3	4	5	6	7	8	v^*
x_1	1	—	—	—	—	—	—	—	—	—
x_2	6	—	—	—	—	—	—	—	—	—
x_3	—	2	—	—	—	—	—	—	—	—
x_4	—	3	—	—	—	—	—	—	—	—
x_5	—	—	v^*	—	—	—	—	v^*	—	—
x_6	—	—	—	4	—	—	—	—	—	—
x_7	—	—	—	5	—	—	—	—	—	—
x_8	—	—	—	—	v^*	—	—	—	—	—
x_9	—	—	—	—	—	v^*	—	—	—	—
x_{10}	—	—	—	—	—	—	7	—	—	—
x_{11}	—	—	—	—	—	—	8	—	—	—
x_{12}	—	—	—	—	—	—	—	—	v^*	—

Символ “—” в таблицях переходів означає невизначений стан.

Таблиця переходів у першому випадку має 120 елементів, а в другому – 20.

Умовимося надалі називати *термом t ацикличним*, якщо відповідний йому граф $G_t^h(v_0)$ ацикличний. Ясно, що коли граф ацикличний, то автомат який йому відповідає, теж ацикличний.

З теореми 6.3.2, наслідку 6.4.3 і наведеного зауваження випливає справедливість таких тверджень.

Теорема 10.3.3. Зведення загальних підвиразів терма t виконується алгоритмом МАА за час $O(m)$, де m – число дуг у графі, який представляє терм t .

Теорема 10.3.4. Зведення загальних підвиразів ацикличного терма t з урахуванням комутативності операцій із Ω виконується алгоритмом АКЗАА за час $O(k \cdot m)$, де $k = |X|$, m – число дуг у графі, який представляє терм t .

Практична цінність наведених теорем полягає в тому, що вони дають спосіб представлення термів у пам'яті ЕОМ, використовуючи найменший її об'єм.

Алгебру $T(X, \Omega)$ будемо називати *A-алгеброю* (*K-алгеброю*), якщо в множині Ω є деяка непуста підмножина Ω_A (Ω_K) асо-

ціативних (комутативних) операцій. При цьому $T(X, \Omega)$ будемо називати **$A\mathbf{K}$ -алгеброю**, якщо $\Omega_A = W_K \neq \emptyset$. Абсолютно вільну алгебру $T(X, \Omega)$ будемо називати **0-алгеброю**. Нехай Eq — скінчена множина рівностей $t_1 = t'_1, \dots, t_n = t'_n$. Вираз $t = t'(\text{mod } Eq)$ означає, що терм t можна перетворити в терм t' , користуючись лише рівностями із Eq .

З теорем 6.4.1, 6.4.2, 10.3.1 і 10.3.2 випливає справедливість таких тверджень.

Теорема 10.3.4. Зведення загальних підвиразів терма t виконується алгоритмами УАКЗ і УАКСЗ в:

0-алгебрі за час $O((\log r) \cdot m \cdot \log((\log r) \cdot m))$;

K -алгебрі за час $O(r \cdot m \cdot \log n)$, де r — максимальна арність комутативної операції із Ω .

Теорема 10.3.5. Зведення загальних підвиразів ацикличного терма t виконується алгоритмом АКЗАА в:

0-алгебрі за час $O(m)$;

K -алгебрі за час $O(r \cdot m)$, де r — максимальна арність комутативної операції із Ω .

Нехай $t_1, t_2, \dots, t_n \in T(X, \Omega)$ і $G_g(V_i, E_i)$ — граф, який представляє терм t_i , $i = 1, 2, \dots, n$.

Теорема 10.3.6. Перевірка тотожності скінченного числа термів t_1, t_2, \dots, t_p виконується алгоритмом УАКЗ в:

0- і A -алгебрах за час $O(M \cdot \log M)$;

K - і AK -алгебрах за час $(r \cdot M \cdot \log M)$, де

$$M = \sum_{i=1}^p |E_i|.$$

Д о в е д е н н я. У першому випадку перетворимо терм t так, щоб дужки у виразах, які містять асоціативні операції, були згруповані, наприклад, праворуч. Після цього можна застосувати алгоритм МАА.

У другому випадку переводимо до так званого асоціативного графа (рис. 10.3.4), що відповідає введенню однієї і тієї операції з різними арностями. Після цього застосовується алгоритм АКЗАА.

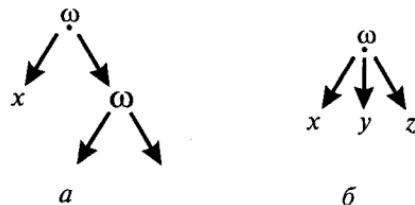


Рис. 10.3.4.
Граф терма $t(a)$ і асоціативний граф терма $t(b)$

Теорема 10.3.7. Перевірка тотожності скінченного числа ациклических термів t_1, t_2, \dots, t_p виконується алгоритмом АКЗАА в:

0- і A -алгебрах за час $O(M)$;

K- і АК-алгебрах за час ($r \cdot M$), де

$$M = \sum_{i=1}^r |E_i|.$$

Доведення аналогічне доведенню попередньої теореми.

6.2. ЗАДАННЯ СКІНЧЕННИХ АВТОМАТІВ ЗА ДОПОМОГОЮ СИСТЕМИ ПРАВИЛ ПЕРЕПИСУВАННЯ

Останнім часом широкого використання набули системи для виконання різного роду як символічних, так і аналітичних перетворень виразів у різних алгебрах. Робота цих систем зводиться до того, що вирази в тій чи іншій алгебрі перетворюються згідно з заданою системою рівностей (правил переписування) цієї алгебри. Однією з перших систем такого роду була система АНАЛІТИК, розроблена в Інституті кібернетики ім. В.М.Глушкова НАН України ще в 1967 р. Серед сучасних таких систем найпопулярнішими є системи REDUCE, AXIOM, APS-1 (остання розроблена в Інституті кібернетики НАН України) та ін. Роботу систем правил переписування розглянемо на прикладі системи APS-1.

В системі алгебраїчного програмування APS-1 вхідні дані і системи правил переписування для перетворення цих даних зображаються у вигляді дерев. Причому при побудові дерева для виразу, який включає асоціативні операції, в цій системі вибрана правостороння орієнтація дужок. Мова програмування цієї системи — APLAN [80], система правил переписування в якій задається у вигляді

$$u \rightarrow (p = q),$$

де u — умова застосування правила $p = q$ з традиційною семантикою: якщо умова u виконується, то застосовується правило $p = q$ (тобто ліва частина p замінюється правою частиною q), інакше це правило не застосовується. Якщо умова в правилі відсутня, то в цьому випадку вважається, що вона тотожно істинна.

Нехай $A = (A, X, f, a_0, F)$ — скінчений автомат без виходів, заданий за допомогою таблиці переходів trs.

trs	a_0	a_1	...	a_n
x_1	a_{11}	a_{12}	...	a_{1n}
x_2	a_{21}	a_{22}	...	a_{2n}
.
.
.
x_m	a_{m1}	a_{m2}	...	a_{mn}

Таке задання легко трансформувати в систему правил переписування, яка буде виконувати роботу автомата A . Побудова системи правил переписування за таблицею переходів зводиться до такого процесу. Починаючи зі стану a_0 , переходи $f(a_i, x_k) = a_j$ замінюються правилами:

якщо стан $a_j \in F$, то генерується два правила:

$$(a_i, x \cdot y) = \text{trs}(a_j, y),$$

$$(a_i, x) = 1 \text{ або довільний інший символ};$$

якщо ж $a_j \notin F$, то генерується єдине правило:

$$(a_i, x \cdot y) = \text{trs}(a_j, y).$$

Побудувавши таким чином всі правила за таблицею переходів, процес генерації правил завершується доповненням правила

$$(x, y) = 0.$$

Очевидно, що таке задання автомата адекватно заданню автомата таблицею переходів.

Перейдемо тепер до розгляду деяких задач з теорії вільних напівгруп, груп і скінченних алгебр.

6.3. ЗАДАЧІ З ТЕОРІЇ СКІНЧЕННОПОРДЖЕНИХ ВІЛЬНИХ ГРУП

Скінчена послідовність P слів вільної групи визначає деяку підгрупу H вільної групи $F(X)$. Послідовність P називається *системою твірних підгрупи H* , якщо слова, з яких складається послідовність P , породжують цю підгрупу. Якщо P є системою твірних для підгрупи H , то ця підгрупа визначається системою P однозначно. Систему P твірних групи H називають *базисом* (системою вільних твірних) цієї групи, якщо кожне слово із H можна представити єдиним способом у вигляді добутку твірних системи P , тобто рівність

$$p_{i_1}^{l_1} \cdots p_{i_k}^{l_k} = p_{j_1}^{q_1} \cdots p_{j_s}^{q_s},$$

де $l_i, q_j = \pm 1$, можлива тоді і тільки тоді, коли

$$k = s, i_1 = j_1, \dots, i_k = j_s, l_1 = q_1, \dots, l_k = q_s.$$

Найвідоміші базиси підгруп вільних груп — нільсенівські системи твірних [49].

Нехай $P = \{p_1, \dots, p_m\}$ — система твірних групи H . Слово $q \in F(X)$ називається *ізольованим* відносно системи P , якщо існує не більше одного слова p_i із P , такого, що q є початком (кінцем) або для p_i , або для p_i^{-1} ($i = 1, 2, \dots, m$). Старшим початком

(кінцем) слова p називається такий початок (кінець) q цього слова, довжина якого задовільняє нерівності

$$1/2 \cdot l(p) < l(q) \leq 1/2 \cdot l(p) + 1.$$

Для слів парної довжини звичайним чином визначається їх ліва і права половини.

Послідовність $P = \{p_1, p_2, \dots, p_m\}$ непустих нескорочуваних слів називається **нільсенівською**, якщо виконуються такі дві умови:

- старші початки і кінці всіх слів із P ізольовані;
- для кожного слова із P парної довжини хоча б одна з його половин — ліва або права — ізольована [44].

Ефективний алгоритм побудови нільсенівської системи твірних можна знайти в [44] разом з його обґрунтуванням. Тут ми обмежимося лише остаточною версією цього алгоритму і оцінкою його часової складності.

Нехай P — початкова система твірних, $\text{Con}(P)$ — множина всіх наслідків, які можна одержати із системи P однократним застосуванням нільсенівських перетворень:

а) заміною p_i із P результатом скорочення будь-якого із добутків $p_i p_j^{-1}$, $p_j^{-1} p_i$, $p_i p_j$, $p_j p_i$, $p_i, p_j \in P$, $i \neq j$;

б) перестановкою слів і заміною слова p_i йому оберненим p_i^{-1} ($i = 1, 2, \dots, m$) в системі P .

Нехай предикат $\Pi(P)$ виконується тоді і тільки тоді, коли праві (або ліві) половини всіх слів парної довжини ізольовані в P , а $P(i)$ означає i -й елемент послідовності P і $L(P(i))$ — його довжину.

При таких домовленостях алгоритм Нільсена набуває такого вигляду.

АЛГОРИТМ НІЛ(P_0)

пачаток.

$P := P_0;$

ІС: якщо $(\exists P' \in \text{Con}(P)) (L(P) > L(P'))$ то $(P := P', \text{на IC}).$

якщо $\Pi(P)$ то **стоп.**

Упорядкувати слова P за зростанням їх довжини і помістити пусті слова в кінець послідовності P .

для $i = 2, 3, \dots, m$ **виконати**

пачаток.

якщо $L(P(i)) = 0$ то **на ІС.**

ОБЕРНЕНИЙ := 0.

ПОВТ: **для** $j = 1, 2, \dots, i - 1$ **виконати**
пачаток.

якщо $L(P(i)) > L(P(i) \cdot P(j)^{-1})$ то $(P(i) := P(i) \cdot P(j)^{-1})$
на ІС).

якщо $L(P(i)) = L(P(i) \cdot P(j)^{-1})$ то $P(i) := P(i) \cdot P(j)^{-1}$.

кінець.

якщо ОБЕРНЕНИЙ = 0 то (ОБЕРНЕНИЙ := 1, $P(i) := P(i)^{-1}$,
на ПОВТ).

якщо $L(P(i))$ парне то

для $j = 1, 2, \dots, i - 1$ виконати
початок.

якщо $L(P(j)) > L(P(i) \cdot P(j))$ то $(P(j) := P(i) \cdot P(j))$
на ІС).

якщо $L(P(j)) = L(P(i) \cdot P(j))$ то $P(j) := P(i) \cdot P(j)$.

кінець.

кінець.

на ІС.

кінець.

Відомо, що часова складність цього алгоритму пропорційна величині $|P_0| \cdot L(P_0)^2$ [44].

Перейдемо тепер до розгляду деяких задач теорії вільних груп і методів їх розв'язання.

Задача включення для слів вільної групи: дана скінчenna послідовність слів $P = \{p_1, \dots, p_m\}$ вільної групи $F(X)$ і деяке слово $q \in F(X)$. Необхідно визначити, належить чи ні слово q до підгрупи H , яка породжена послідовністю слів P .

Не обмежуючи загальності, будемо вважати, що система твірних P є базисом (нільсенівською системою твірних) (якщо це не так, то, застосовуючи до системи P алгоритм побудови нільсенівської системи твірних, одержимо такий базис).

Розв'язати задачу включення для слів можна двома шляхами.

Перший спосіб розв'язання випливає з такого твердження.

Теорема 10.3.8. *Підгрупа вільної групи скінченнопороджена тоді і тільки тоді, коли множина її нескорочуваних слів є регулярною множиною* [3, 35].

З цієї теореми випливає існування скінченного автомата A , який представляє нескорочувані слова підгрупи H деякою множиною станів F . Відомо, що часова складність алгоритму побудови автомата A за нільсенівською системою твірних $P = \{p_1, \dots, p_m\}$ пропорційна величині $|P| \cdot L(P)$ [35].

Звідси випливає алгоритм розв'язання даної задачі. Дійсно, побудувавши за множиною P автомат A , який представляє нескорочувані слова підгрупи H , подамо на його вход слово q . Якщо стан $a = f(a_0, q) \in F$, то $q \in H$, у протилежному випадку $q \notin H$.

Другим шляхом розв'язання задачі включення слова q в H є зведення слова q до пустого слова e за допомогою множення

слова q на базисні елементи групи H . Якщо $q \in H$, то q вирахується через базисні елементи i , значить, існує такий базисний елемент p , що довжина одного із слів rq , qp , $p^{-1}q$, qp^{-1} буде не більшою за довжину слова q (внаслідок нільсенівської властивості базисних елементів). Виконуючи заміну слова q словом меншої довжини і застосовуючи таке множення до нового слова, ми врешті-решт одержимо або пусте слово e , або слово p' , довжина якого не зменшується при черговому множенні на базисний елемент. Якщо в результаті одержали слово e , то $q \in H$, інакше $q \notin H$.

Зауважимо, що перший шлях розв'язання цієї проблеми ефективніший за другий. Часові характеристики відрізняються більше ніж на порядок.

Задача включення для підгруп: за двома підгрупами H_1 і H_2 , заданими системами своїх твірних P_1 і P_2 відповідно, визначити включення $H_1 \subseteq H_2$ або $H_2 \subseteq H_1$.

Розв'язання цієї задачі зводиться до побудови нільсенівських базисів для H_1 і H_2 за множинами P_1 і P_2 . А потім, використовуючи розв'язок задачі включення для слів, можна визначити включення кожного елемента базису підгрупи H_1 (H_2) в підгрупу H_2 (H_1). *

З розв'язання цієї задачі очевидним чином випливає розв'язання **задачі тотожності для підгруп:** чи породжують скінченні системи твірних P_1 і P_2 одну і ту ж підгрупу вільної групи чи ні?

Задача ізоморфізму для підгруп вільних груп: визначити за двома заданими скінченнопородженими підгрупами H_1 і H_2 вільних груп $F(X)$ і $F(Y)$ відповідно, чи ізоморфні H_1 і H_2 чи ні?

Розв'язання цієї задачі зводиться до побудови нільсенівської системи твірних для H_1 і H_2 та порівняння потужностей одержаних базисів. Якщо ці потужності рівні, то підгрупи ізоморфні, в протилежному випадку — неізоморфні. Цей факт випливає з такого твердження.

Теорема 10.3.9. *Вільні групи ізоморфні тоді і тільки тоді, коли множини їх вільних твірних рівнопотужні.*

Ця теорема є наслідком загальнішої теореми про те, що ранг вільної групи не залежить від вибору системи вільних твірних цієї групи [40, стор. 220].

На завершення цього підрозділу зауважимо, що все сказане про вільні групи справедливе і по відношенню до скінченнопороджених вільних напівгруп. При цьому деякі побудови спрощуються. Це стосується насамперед побудови автомата, який представляє слова напівгрупи. У цьому випадку алгоритм побудови автомата за системою твірних простіший, оскільки немає скорочення елементів.

Приклад 10.3.5

Розглянемо напівгрупу, породжену елементами $P = \{aa, ab, ac, cd, da, dc\}$. За базисом P будуємо систему правил переписування trs , яка реалізує таблицю переходів автомата A (автомата, який представляє напівгрупу, породжену базисом P).

Побудова системи правил trs виконується за два етапи:

1) спочатку базис P представляється в так званому приведеному вигляді, тобто у вигляді, де у виразі $P = p_1 \vee p_2 \vee \dots \vee p_k$ винесені за дужки спільні множники (в нашому прикладі $P = a \cdot (a \vee b) \vee c \cdot (c \vee d) \vee d \cdot (a \vee c)$);

2) потім за отриманим базисом P будується система trs , яка реалізує функцію переходів автомата, що представляє слова множини P .

В результаті одержуємо таку систему trs :

$$\begin{aligned} \text{trs} := & rs(x, y) ((0, a \cdot y) = \text{trs}(2, y), \\ & (0, c \cdot y) = \text{trs}(1, y), \\ & (0, d \cdot y) = \text{trs}(3, y), \\ & (1, d \cdot y) = \text{trs}(0, y), \quad (1, d) = 4, \\ & (1, c \cdot y) = \text{trs}(0, y), \quad (1, c) = 4, \\ & (2, a \cdot y) = \text{trs}(0, y), \quad (2, a) = 4, \\ & (2, b \cdot y) = \text{trs}(0, y), \quad (2, b) = 4, \\ & (3, a \cdot y) = \text{trs}(0, y), \quad (3, a) = 4, \\ & (3, c \cdot y) = \text{trs}(0, y), \quad (3, c) = 4, \\ & (x, y) = 5. \end{aligned}$$

При цьому слово p належить даній напівгрупі, якщо автомат зупиняється в стані 4, і не належить, якщо автомат зупиняється в стані 5. ▲

6.4. СКІНЧЕННІ АЛГЕБРИ

Оскільки скінченні алгебри мають скінченне число елементів, то вони є регулярними множинами і, значить, можуть бути представлені в скінченних автоматах деякою множиною станів. Обмежимося розглядом скінченної алгебри G з однією бінарною операцією, заданої таблицею Келі:

ω	a_0	a_1	...	a_n
a_0	a_{00}	a_{01}	...	a_{0n}
a_1	a_{10}	a_{11}	...	a_{1n}
...
...
...
a_n	a_{n0}	a_{n1}	...	a_{nn}

Таблицю Келі можна вважати таблицею переходів автомата A , який представляє алгебру G . При цьому множина станів автомата збігається з множиною заключних станів F , а функція переходів f задається рівнянням

$$f(a_i, a_j) = \omega(a_i, a_j), \quad i, j = 0, 1, 2, \dots, n.$$

Приклад 10.3.6

Нехай задана скінчена абелева група своєю таблицею додавання.

Таблиця

+	0	1	2	3	4
0	0	1	2	3	4
1	1	4	3	0	2
2	2	3	1	4	0
3	3	0	4	2	1
4	4	2	0	1	3

Про комутативність даної групи свідчить симетричність таблиці додавання. Відповідна система правил переписування видає результат (значення виразу) або ж знак помилки err, якщо у виразі є елемент, для якого не визначена операція додавання:

$$\begin{aligned}
 \text{trsa} := & rs(x, y) ((0, 0 + y) = \text{trsa}(0, y), (0, 0) = 0, \\
 & (0, 1 + y) = \text{trsa}(1, y), (0, 1) = 1, \\
 & (0, 2 + y) = \text{trsa}(2, y), (0, 2) = 2, \\
 & (0, 3 + y) = \text{trsa}(3, y), (0, 3) = 3, \\
 & (0, 4 + y) = \text{trsa}(4, y), (0, 4) = 4, \\
 & (1, 0 + y) = \text{trsa}(1, y), (1, 0) = 1, \\
 & (1, 1 + y) = \text{trsa}(4, y), (1, 1) = 4, \\
 & (1, 2 + y) = \text{trsa}(3, y), (1, 2) = 3, \\
 & (1, 3 + y) = \text{trsa}(0, y), (1, 3) = 0, \\
 & (1, 4 + y) = \text{trsa}(2, y), (1, 4) = 2, \\
 & (2, 0 + y) = \text{trsa}(2, y), (2, 0) = 2, \\
 & (2, 1 + y) = \text{trsa}(3, y), (2, 1) = 3, \\
 & (2, 2 + y) = \text{trsa}(1, y), (2, 2) = 1, \\
 & (2, 3 + y) = \text{trsa}(4, y), (2, 3) = 4, \\
 & (2, 4 + y) = \text{trsa}(0, y), (2, 4) = 0, \\
 & (3, 0 + y) = \text{trsa}(3, y), (3, 0) = 3, \\
 & (3, 1 + y) = \text{trsa}(0, y), (3, 1) = 0, \\
 & (3, 2 + y) = \text{trsa}(4, y), (3, 2) = 4, \\
 & (3, 3 + y) = \text{trsa}(2, y), (3, 3) = 2, \\
 & (3, 4 + y) = \text{trsa}(1, y), (3, 4) = 1, \\
 & (4, 0 + y) = \text{trsa}(4, y), (4, 0) = 4, \\
 & (4, 1 + y) = \text{trsa}(2, y), (4, 1) = 2, \\
 & (4, 2 + y) = \text{trsa}(0, y), (4, 2) = 0, \\
 & (4, 3 + y) = \text{trsa}(1, y), (4, 3) = 1, \\
 & (4, 4 + y) = \text{trsa}(3, y), (4, 4) = 3, \\
 & (x, y) = \text{err}).
 \end{aligned}$$

Систему правил переписування в тому чи іншому випадку можна записати економнішим способом (зводячи до мінімуму число правил). Наприклад, враховуючи властивість нульового елемента, наведену систему правил можна записати більш економно:

$$\begin{aligned}
 \text{trsa} := & rs(x, y) ((x, 0 + y) = \text{trsa}(x, y), \\
 & (0, x + y) = \text{trsa}(x, y), \quad (0, x) = x, \\
 & (1, 1 + y) = \text{trsa}(4, y), \quad (1, 1) = 4, \\
 & (1, 2 + y) = \text{trsa}(3, y), \quad (1, 2) = 3, \\
 & (1, 3 + y) = \text{trsa}(0, y), \quad (1, 3) = 0, \\
 & (1, 4 + y) = \text{trsa}(2, y), \quad (1, 4) = 2, \\
 & (2, 1 + y) = \text{trsa}(3, y), \quad (2, 1) = 3, \\
 & (2, 2 + y) = \text{trsa}(1, y), \quad (2, 2) = 1, \\
 & (2, 3 + y) = \text{trsa}(4, y), \quad (2, 3) = 4, \\
 & (2, 4 + y) = \text{trsa}(0, y), \quad (2, 4) = 0, \\
 & (3, 1 + y) = \text{trsa}(0, y), \quad (3, 1) = 0, \\
 & (3, 2 + y) = \text{trsa}(4, y), \quad (3, 2) = 4, \\
 & (3, 3 + y) = \text{trsa}(2, y), \quad (3, 3) = 2, \\
 & (3, 4 + y) = \text{trsa}(1, y), \quad (3, 4) = 1, \\
 & (4, 1 + y) = \text{trsa}(2, y), \quad (4, 1) = 2, \\
 & (4, 2 + y) = \text{trsa}(0, y), \quad (4, 2) = 0, \\
 & (4, 3 + y) = \text{trsa}(1, y), \quad (4, 3) = 1, \\
 & (4, 4 + y) = \text{trsa}(3, y), \quad (4, 4) = 3, \\
 & (x, y) = \text{err}). \quad \blacksquare
 \end{aligned}$$

Розглянемо тепер представлення скінченної алгебри, заданої двома таблицями Келі (наприклад, для множення і для додавання). Діючи аналогічно тому, як це робилося вище, будуємо дві системи правил переписування, а потім добавляємо ще одну систему правил, яка зв'язує обидві системи в одне ціле.

Приклад 10.3.7

Розглянемо кільце, адитивною групою якого є група з прикладу 10.3.6, а операція множення задана таблицею.

Згідно з описаним вище, система правил переписування для операції множення має вигляд (ця система оптимізована з урахуванням властивостей констант 0 і 1 в кільці):

$$\begin{aligned}
 \text{trsm} := & rs(x, y) ((0, x) = 0, \\
 & (x, 0) = 0, \quad (x, 0 \cdot y) = 0, \\
 & (1, x \cdot y) = \text{trsm}(x, y), \quad (1, x) = x,
 \end{aligned}$$

Таблиця

.	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	3	4	1
3	0	3	4	1	2
4	0	4	1	2	3

$$\begin{aligned}
(x, 1 \cdot y) &= \text{trsm}(x, y), & (x, 1) &= x, \\
(2, 2 \cdot y) &= \text{trsm}(3, y), & (2, 2) &= 3, \\
(2, 3 \cdot y) &= \text{trsm}(4, y), & (2, 3) &= 4, \\
(2, 4 \cdot y) &= \text{trsm}(1, y), & (2, 4) &= 1, \\
(3, 2 \cdot y) &= \text{trsm}(4, y), & (3, 2) &= 4, \\
(3, 3 \cdot y) &= \text{trsm}(1, y), & (3, 3) &= 1, \\
(3, 4 \cdot y) &= \text{trsm}(2, y), & (3, 4) &= 2, \\
(4, 2 \cdot y) &= \text{trsm}(1, y), & (4, 2) &= 1, \\
(4, 3 \cdot y) &= \text{trsm}(2, y), & (4, 3) &= 2, \\
(4, 4 \cdot y) &= \text{trsm}(3, y), & (4, 4) &= 3, \\
(x, y) &= \text{err1}).
\end{aligned}$$

Перетворення двох систем правил переписування trsa і trsm в кільце виконується за допомогою простої системи правил переписування, яка дозволяє виконувати обчислення виразів у цьому кільці.

$$\begin{aligned}
\text{ring} := & rs(x, y) ((x + y) = \text{trsa}(\text{ring}(x), \text{ring}(y)), \\
& (x \cdot y) = \text{trsm}(\text{ring}(x), \text{ring}(y))), \\
\text{atom}(x)(x = x), & \\
x = \text{err2}).
\end{aligned}$$

У наведений системі правил використовується підсистема, яка визначає, чи є даний вираз числом.

$$\begin{aligned}
\text{atom} := & rs(x, y) (x + y = 0, \\
& x \cdot y = 0, \\
& x = 1).
\end{aligned}$$

Алгоритм побудови такого роду системи правил переписування зводиться до такої послідовності дій. Визначається головна операція при обчисленні значення виразу, і в залежності від цієї операції в правій частині правила переписування викликається система правил для обчислення значення цієї операції.

Задачі і вправи

- Перевірте, користуючись процедурою Кнута–Бендікса, чи буде канонічною множина тотожностей теорії вільних напівгруп, груп і кілець.
- Представте скінченним автоматом такі терми:

$$\begin{aligned}
 t_1 &= (((x_1 \cdot x_2) / (x_2 \cdot x_3)) - (x_1 \cdot x_2)), \\
 t_2 &= (((x_1 + x_2) - (x_1 \cdot x_2)) / (x_1 + x_2)), \\
 t_3 &= (((x_1 - (x_2 \cdot x_1)) + x_2) - (x_2 + x_1)).
 \end{aligned}$$
- Побудуйте приведений автомат для автомата із задачі 1.
- Застосовуючи алгоритм УАКСЗ до автомата із задачі 1, знайдіть симетричне конгруентне замикання відношення i — відношення тотожності, якщо операції $+$ і \cdot вважаються комутативними.

5. Побудуйте автомат, який представляє множину слів: конституція, конструкція, конструктор, конституента, конкатенація, контроль.

Побудуйте приведений автомат до побудованого і переконайтесь, що він займає менше місця, ніж початковий автомат.

6. Побудуйте конгруентне замикання відношення R , яке задається такими співвідношеннями:

$$x_1 \cdot x_2 = x_1 - x_2, x_2 - x_1 = x_1 - x_2$$

для автомата, який представляє терми

$$\begin{aligned}t_1 &= (((x_1 - x_2) + x_3) \cdot (x_2 - x_1)) / (x_1 - x_3), \\t_2 &= (((x_1 \cdot x_2) + (x_2 \cdot x_1)) - (x_2 \cdot x_1)).\end{aligned}$$

7. Побудуйте систему правил переписування для побудови канонічної форми елементів у теорії вільних абелевих груп.

§ 10.4. ТЕОРЕМА ХОЛЛА ТА ЇЇ ЗАСТОСУВАННЯ

У даному параграфі розглядаються важливі застосування методів теорії графів, пов'язані з потоками в мережах. Теоретичною базою розв'язання задачі про максимальний потік при мінімальному розрізі є дві важливі теореми: Холла і Менгера.

Розглянемо задачу комбінаторного характеру, яка відома під назвою *задачі про весілля*.

Дано дві скінченні множини A ($|A| = m$) — множина юнаків і B ($|B| = n$) — множина дівчат, причому кожен юнак знайомий з кількома дівчатаами. За яких умов можна одружитися хлопцям так, щоб кожен з них одружився із знайомою дівчиною? (Будемо вважати, що полігамія не дозволяється).

Приклад 10.4.1

Нехай $A = \{a_1, a_2, a_3, a_4\}$ і $B = \{b_1, b_2, b_3, b_4, b_5\}$, а відношення знайомства між хлопцями та дівчатаами показані в таблиці.

Можливий розв'язок задачі такий: (a_1, b_4) , (a_2, b_1) , (a_3, b_3) , (a_4, b_2) .

Таблиця

Юнаки	Знайомі їм дівчата
a_1	b_1, b_4, b_5
a_2	b_1
a_3	b_2, b_3, b_4
a_4	b_2, b_4

Розглянуту задачу можна представити графічно за допомогою двочастинного графа $G = (V, E)$, де $V = A \cup B$, а $E = \{(a_1, b_1), (a_1, b_4), (a_1, b_5), (a_2, b_1), (a_3, b_2), (a_3, b_3), (a_3, b_4), (a_4, b_2), (a_4, b_4)\}$ (див. граф на рис. 10.4.1).

Для того щоб точно сформулювати задачу про весілля в термінах теорії графів, введемо таке поняття.

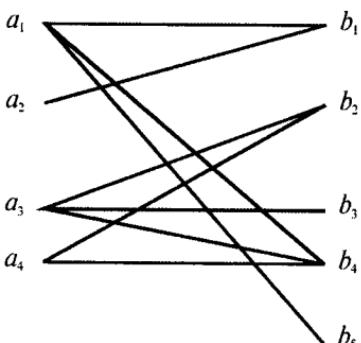


Рис. 10.4.1

Нехай $G = (A \cup B, E)$ — двочастинний граф, побудований вище. **Ідеальним паросполученням** із A в B у двочастинному графі G називається така взаємно однозначна відповідність між вершинами із A і підмножиною вершин із B , при якій відповідні вершини суміжні, тобто з'єднані ребром. Тепер задачу про весілля в термінах теорії графів можна сформулювати так: якщо $G = (A \cup B, E)$ — двочастинний граф, то при яких умовах в G існує ідеальне паросполучення з A в B ? Відповідь на це питання дає знаменита теорема Холла.

Теорема Холла про весілля. Нехай A ($|A| = m$) — множина юнаків і B ($|B| = n$) — множина дівчат, причому кожний з юнаків знайомий з кількома дівчатаами. Задача про весілля має розв'язок тоді і тільки тоді, коли будь-які k юнаків з множини A знайомі в сукупності не менш ніж з k дівчатаами з множини B ($1 \leq k \leq m$).

Доведення. Необхідність у даному випадку випливає з того, що коли дана умова не виконується для якої-небудь множини із k юнаків, то одружитися потрібним способом не можуть навіть ці k юнаків, не кажучи вже про решту.

Достатність будемо доводити індукцією за числом юнаків m . Ясно, що при $m = 1$ теорема справедлива. Припустимо, що число юнаків $m > 1$. Розглянемо два випадки.

1. Припустимо, що кожні k юнаків знайомі з $k + 1$ дівчиною. Тоді, якщо кожний юнак одружиться зі знайомою йому дівчиною, то для решти $m - k$ юнаків залишається справедливою початкова умова. За припущенням індукції ці $m - k$ юнаків можуть одружитися вказаним способом, і тим самим теорема в цьому випадку буде доведена.

2. Нехай тепер k юнаків ($k < m$) знайомі з k дівчатаами, і вони можуть одружитися потрібним способом. Серед решти $m - k$ юнаків кожні h із них повинні бути знайомі не менш ніж з h дівчатаами, оскільки в протилежному випадку ці h юнаків разом з вибраними раніше k юнаками будуть знайомі менш ніж з $h + k$ дівчатаами, а це суперечить нашій умові. Отже, для $m - k$ юнаків виконується початкова умова і, за припущенням індукції, $m - k$ юнаків можуть одружитися так, як це вимагається.

Теорема доведена.

Наслідок 10.4.1. Нехай $G = (A \cup B, E)$ — двочастинний граф і для всякої підмножини A' із A $f(A') = \{b \in B \mid (a, b) \in E, a \in A'\}$. В

такому разі ідеальне паросполучення із A в B існує тоді і тільки тоді, коли $(\forall A' \subseteq A) |A'| \leq |f(A)|$.

Розглянемо деякі застосування теореми Холла.

1. ТРАНСВЕРСАЛІ

Нехай E — непуста скінчenna множина і $S = \{S_1, S_2, \dots, S_m\}$ — сімейство не обов'язково різних непустих її підмножин. *Трансверсаллю*, або *системою різних представників*, для S називається підмножина множини E , яка складається з m різних елементів: по одному з кожної множини S_i , $i = 1, 2, \dots, m$.

Приклад 10.4.2

Повертаючись до прикладу 10.4.1 з юнаками і дівчатами, ми побудували трансверсалль для множини $S = \{S_1, S_2, S_3, S_4\}$, де $S_1 = \{b_1, b_4, b_5\}$, $S_2 = \{b_1\}$, $S_3 = \{b_2, b_3, b_4\}$, $S_4 = \{b_2, b_4\}$. Цією трансверсаллю була множина $T = \{b_4, b_1, b_3, b_2\}$. *

Розглянемо інший приклад. Нехай $E = \{1, 2, 3, 4, 5, 6\}$, а $S = \{S_1, S_2, S_3, S_4, S_5\}$, де $S_1 = S_2 = \{1, 2\}$, $S_3 = S_4 = \{2, 3\}$, $S_5 = \{1, 4, 5, 6\}$. Сімейство S не має трансверсалі, оскільки неможливо знайти п'ять різних елементів із E — по одному зожної із підмножин S_1, S_2, S_3, S_4, S_5 . Слід зауважити, що його підсімейство $S' = \{S_1, S_2, S_3, S_5\}$ має трансверсалль, наприклад $\{1, 2, 3, 4\}$. Трансверсалль довільного підсімейства сімейства S називається *частковою трансверсаллю* для S . Часткових трансверсалей для даного сімейства S може бути кілька. В нашому прикладі сімейство S має такі часткові трансверсалі: $\{1, 2, 3, 6\}$, $\{2, 3, 6\}$, $\{1, 5\}$, \emptyset тощо. Неважко зрозуміти, що всяка підмножина часткової трансверсалі сама буде частковою трансверсаллю.

Виникає питання: при яких умовах задане сімейство підмножин деякої множини має трансверсалль.

Поставлене питання легко пов'язати із задачею про весілля. Для цього достатньо взяти за E множину дівчат, а за S_i — множину дівчат, які знайомі з a_i -м юнаком ($i = 1, 2, \dots, m$). Трансверсаллю в цьому випадку буде множина з m дівчат, така, що кожному юнакові відповідає лише одна знайома йому дівчина. Отже, теорема Холла дає критерій існування трансверсалі для заданого сімейства множин. Формульовання теореми Холла для цього випадку має такий вигляд.

Теорема Холла для трансверсалей. Нехай E — непуста скінчена множина і $S = \{S_1, S_2, \dots, S_m\}$ — сімейство не обов'язково різних непустих її підмножин. У такому разі S має трансверсалль тоді і

тільки тоді, коли для будь-яких k підмножин S_i їх об'єднання містить не менше, ніж k різних елементів ($k = 1, 2, \dots, m$).

Розглянемо тепер питання про спільні трансверсалі. Нехай E — непуста скінчена множина, а $S = \{S_1, S_2, \dots, S_m\}$ і $T = \{T_1, T_2, \dots, T_m\}$ — два сімейства її непустих підмножин. Цікаво знати, коли існує спільна трансверсаль для S і T , тобто така множина, яка складається із m різних елементів множини E і яка є трансверсаллю і для S , і для T .

До задачі про спільну трансверсаль зводиться, наприклад, задача складання розкладу занять. Нехай E — множина відрізків часу, протягом яких можуть читатися ті чи інші лекції; S_i означає множину відрізків часу, протягом яких m професорів бажають читати лекції, а T_i — множина відрізків часу, протягом яких вільні m лекційних аудиторій. Тоді, знайшовши спільну трансверсаль для S і T , ми зможемо виділити кожному професорові вільну аудиторію в зручний для нього час.

Сформулюємо критерій існування спільної трансверсалі двох сімейств S і T .

Теорема 10.4.1. *Нехай E — непуста скінчена множина, а $S = \{S_1, \dots, S_m\}$ і $T = \{T_1, \dots, T_m\}$ — два сімейства її непустих підмножин. Множини S і T мають спільну трансверсаль тоді і тільки тоді, коли для всіх підмножин A і B множини $\{1, 2, \dots, m\}$*

$$\left| \left(\bigcup_{i \in A} S_i \right) \cup \left(\bigcup_{j \in B} T_j \right) \right| \geq |A| + |B| - m.$$

Доведення цієї теореми можна знайти, наприклад, в [68].

Слід зауважити, що умови, при яких існує спільна трансверсаль для трьох сімейств, поки що невідомі.

2. ЛАТИНСЬКІ КВАДРАТИ

Латинським ($m \times n$)-**прямокутником** називається прямокутна матриця $A = (a_{ij})$ розміру $m \times n$, де a_{ij} — натуральні числа, які задовольняють такі умови:

- 1) $1 \leq a_{ij} \leq m$;
- 2) елементи a_{ij} в кожному рядку і в кожному стовпчику різні.

Слід зауважити, що з умов 1), 2) випливає нерівність $m \leq n$. Якщо ж $m = n$, то латинський прямокутник називається **латинським квадратом**. Вияснимо таке питання: якщо заданий латинський $m \times n$ -прямокутник, де $m < n$, то при яких умовах можна приєднати до нього $n - m$ нових рядків так, щоб у резуль-

таті можна було одержати латинський квадрат? Відповідь на це питання дає така теорема.

Теорема 10.4.2. Якщо A — латинський $m \times n$ -прямокутник і $m < n$, то A можна розширити до латинського квадрата шляхом приєднання $n - m$ нових рядків.

Д о в е д е н н я. Опишемо насамперед процедуру, за допомогою якої A можна розширити до латинського $(m + 1) \times n$ -прямокутника, а потім, повторюючи дану процедуру, перейдемо до латинського квадрата.

Нехай $R = \{1, 2, \dots, n\}$ і $S = \{S_1, S_2, \dots, S_n\}$, де S_i означає множину тих елементів із R , які не зустрічаються в i -му рядку матриці A . Необхідно показати, що існує n різних елементів, які входять у кожну із множин S_i , $i = 1, 2, \dots, n$. За теоремою Холла для цього достатньо показати, що об'єднання k будь-яких множин S_i містить не менше, ніж k різних елементів. Але це очевидно, оскільки всяке таке об'єднання має $(n - m) \cdot k$ елементів, враховуючи повторення. Якщо припустити, що в цьому об'єднанні менше, ніж k різних елементів, то це означало б, що хоча б один із них повторюється більше, ніж $n - m$ разів, що неможливо.

Теорема доведена.

§ 10.5. ТЕОРЕМА МЕНГЕРА І ПОТОКИ В МЕРЕЖАХ

Теорема Менгера, яка розглядається нижче, тісно пов'язана з теоремою Холла і в деякому плані еквівалентна їй. Але ця теорема має досить широкі практичні застосування, серед яких знаходиться теорема про *максимальний потік*. Теорема Менгера стосується числа простих шляхів, які з'єднують дві задані вершини u і v графа $G = (V, E)$. Для розв'язання цієї задачі необхідно ввести кілька нових понять.

Нехай l і l' — два простих ланцюги, що з'єднують вершини u і v . Ці ланцюги називаються такими, що *не перетинаються по ребрах*, якщо вони не мають жодного спільного ребра, і такими, що *не перетинаються по вершинах*, якщо вони не мають жодної спільної вершини, зрозуміло, крім вершин u і v . Аналогічні означення можна дати і для орграфів. Надалі, якщо не обумовлене протилежне, будуть розглядатися неорієнтовані графи.

Нехай G — зв'язний граф, а v і w — дві різні його вершини. vw -*Розділюальною множиною* в G називається множина R ребер графа G , така, що всякий простий ланцюг із вершини v у вершину w проходить через ребро з множини R . Аналогічно,

vw-відокремлюальною множиною в G називається множина S його вершин, які не належать вершини v і w та яка має ту властивість, що всякий простий ланцюг із вершини v у вершину w проходить через вершину з множини S .

Приклад 10.5.1

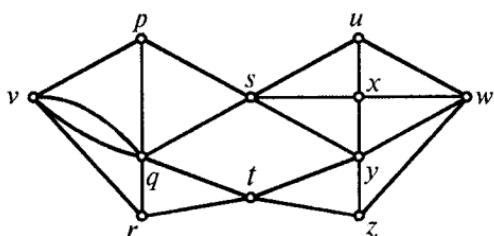


Рис. 10.5.1

Для графа, зображеного на рис. 10.5.1, множини $R = \{(p, s), (q, s), (t, y), (t, z)\}$ і $R' = \{(u, w), (x, w), (y, w), (z, w)\}$ є *vw*-розділюальними, а множини $S = \{p, q, y, z\}$ і $S' = \{s, t\}$ — *vw*-відокремлюальними. ▲

Для підрахунку всіх простих ланцюгів, які не перетинаються по ребрах і які ведуть із вершини v у вершину w , насамперед зауважимо, що коли яка-небудь *vw*-розділюальна множина R має k ребер, то число таких ланцюгів не перевищує k , оскільки в протилежному випадку деяке ребро з R належало б більше, ніж одному простому ланцюгу. Якщо до того ж R є *vw*-розділюальною множиною найменшої можливої потужності, то число простих ланцюгів, що не перетинаються по ребрах, дорівнює k , і, отже, кожний такий ланцюг має рівно одне ребро з R . Цей результат відомий під назвою *реберної теореми Менгера*.

Теорема 10.5.1. *Максимальне число простих ланцюгів, що не перетинаються по ребрах і з'єднують дві різні вершини v і w зв'язного графа G , дорівнює мінімальному числу k ребер у *vw*-розділюальній множині.*

Доведення. З наведеного вище зауваження випливає, що максимальне число простих ланцюгів, що не перетинаються по ребрах і з'єднують вершини v і w , не перевищує мінімального числа ребер у *vw*-розділюальній множині. Покажемо індукцією за числом ребер у графі G , що ці числа рівні.

Нехай G має m ребер. Для $m = 1$ теорема очевидно справедлива. Нехай тепер теорема має місце для всіх графів, число ребер яких менше m . Розглянемо два можливі випадки.

1. Припустимо, що існує *vw*-розділюальна множина R мінімальної потужності k , така, що не всі її ребра інцидентні w . Наприклад, у розглянутому вище графі (рис. 10.5.1) такою множиною є множина $R = \{(p, s), (q, s), (t, y), (t, z)\}$. Після вилучення з графа G ребер, які належать множині R , одержуємо два підграфи V і W , які не мають спільних ребер і

вершин та, наприклад, $v \in V$, а $w \in W$. Побудуємо два нових підграфи G_1 і G_2 таким чином: G_1 — граф, одержаний внаслідок стягування тих ребер графа G , які належать графу V , а G_2 будується аналогічно, тільки ребра стягаються до вершини w . (Графи G_1 , G_2 , одержані з графа G , показані на рис. 10.5.2. Штриховими лініями позначені ребра множини R).

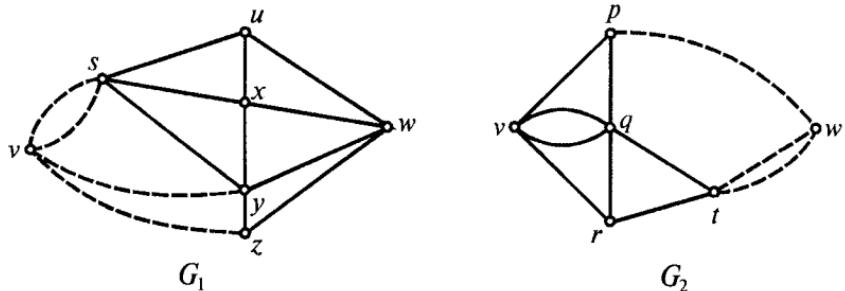


Рис. 10.5.2

Оскільки число ребер в G_1 і G_2 менше, ніж в G , і очевидно, що R — vw -розділювальна множина мінімальної потужності для обох графів G_1 і G_2 , то за припущенням індукції в G_1 існує k простих ланцюгів із v у w , що не перетинаються по ребрах — це справедливо і для графа G_2 . Комбінуючи очевидним способом ці прості ланцюги, одержуємо k шуканих простих ланцюгів у графі G , що не перетинаються по ребрах.

2. Нехай тепер всяка vw -розділювальна множина мінімальної потужності k складається лише з ребер, інцидентних вершині v , або з ребер, інцидентних вершині w . Такою vw -розділювальною множиною буде множина R' для графа G , зображеного на рис. 10.5.1. Не обмежуючи загальності, можна вважати, що кожне ребро графа G належить деякій vw -розділювальній множині потужності k , оскільки в протилежному випадку вилучення відповідного ребра з графа G не впливає на величину k і дає нам право скористатися припущенням індукції, щоб одержати k простих ланцюгів, які не перетинаються по ребрах. Отже, всякий простий ланцюг C із v у w повинен складатися або з одного ребра, або з двох ребер (і тому може включати не більше одного ребра із будь-якої vw -розділювальної множини потужності k). Вилучивши із G ребра, які належать C , одержимо граф, який буде мати принаймні $k - 1$ простий ланцюг, що не перетинаються по ребрах. Разом із C ці прості ланцюги і дають шукані k простих ланцюгів у графі G .

Теорема доведена.

Слід зауважити, що доведення теореми не є конструктивним, оскільки з нього не випливає ні алгоритм знаходження k простих

ланцюгів, що не перетинаються по ребрах, ні навіть спосіб обчислення величини k для заданого графа G . Далі буде наведено алгоритм, який розв'язує обидві ці задачі, а зараз сформулюємо “вершинний” варіант теореми Менгера для графів і “реберний” аналог для орграфів.

Теорема Менгера. *Максимальне число простих ланцюгів, що не перетинаються по вершинах і з'єднують дві різні несуміжні вершини v і w зв'язного графа G , дорівнює мінімальному числу вершин у vw -відокремлювальній множині.*

Теорема 10.5.2. *Максимальне число простих орланцюгів, що не перетинаються по ребрах і з'єднують дві різні вершини v і w орграфа G , дорівнює мінімальному числу дуг у vw -розділювальній множині.*

Доведення обох теорем майже дослівно повторює доведення теореми 10.5.1, і тут вони не наводяться.

Застосуємо теорему 10.5.2 до графа на рис. 10.5.3.

Безпосереднім підрахунком встановлюється, що в цього графа 6 простих ланцюгів із v в w , які не перетинаються по дугах. Дуги, з яких складається відповідна vw -розділювальна множина, позначені штриховими лініями. З цього прикладу випливає, що зростом кількості дуг, які з'єднують пари суміжних вершин, графи стають досить громіздкими. Цю незручність можна обійти, якщо намалювати тільки одну дугу і приписати їй ціле число, яке вказує, скільки таких дуг існує насправді. Наприклад, для графа, зображеного на рис. 10.5.3, маємо граф, зображений на рис. 10.5.4. Це, на перший погляд зовсім незначне перетворення, відіграє важливу роль при вивчені потоків в мережах і транспортних задачах, до розгляду яких ми і переходимо. Насамперед покажемо, що аналіз мереж, по суті, еквівалентний вивченю орграфів. Розглянемо приклади.

Нехай маємо деякий процесор мультипроцесорної ЕОМ, пов'язаний з k іншими процесорами каналами для передачі інформації. При цьому кожний із каналів може пропускати різні об'єми інформації. Припустимо, що передати інформацію від даного процесора його k абонентам можна різними каналами. Необхідно передати максимум інформації за найменший проміжок часу, не перевищуючи допустимих об'ємів інформації, які можуть передаватися через канали.

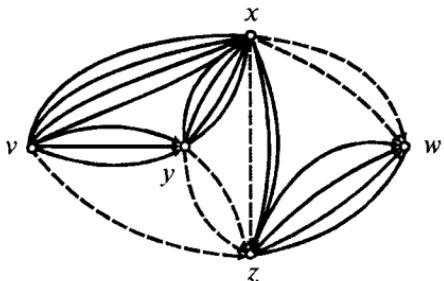


Рис. 10.5.3

позначені штриховими лініями. З цього прикладу випливає, що зростом кількості дуг, які з'єднують пари суміжних вершин, графи стають досить громіздкими. Цю незручність можна обійти, якщо намалювати тільки одну дугу і приписати їй ціле число, яке вказує, скільки таких дуг існує насправді. Наприклад, для графа, зображеного на рис. 10.5.3, маємо граф, зображений на рис. 10.5.4. Це, на перший погляд зовсім незначне перетворення, відіграє важливу роль при вивчені потоків в мережах і транспортних задачах, до розгляду яких ми і переходимо. Насамперед покажемо, що аналіз мереж, по суті, еквівалентний вивченю орграфів. Розглянемо приклади.

Можна навести й інші можливі ситуації. Наприклад, якщо граф, зображений на рис. 10.5.3, інтерпретувати так, що кожна дуга орграфа — вулиця з одностороннім рухом, а числа, що відповідають дугам, означають максимальний можливий потік транспорту (кількість машин за годину) по цій вулиці, то хотілося б знайти найбільше можливе число машин, які можуть проїхати за годину з пункта v в пункт w . Цей орграф можна також розглядати як схему електричного ланцюга, і тоді задача полягає в тому, щоб знайти максимальний струм, який можна пропустити по цьому ланцюгу при умові, що граничні струми окремих проводів ланцюга задані.

Виходячи з даних прикладів, означимо тепер мережу N як орграф, кожній дузі e якого поставлене у відповідність деяке додатне дійсне число $z(e)$. Число $z(e)$ називається *пропускною здатністю дуги e* . Існують й інші, еквівалентні даному, означення мережі: мережа N являє собою пару (G, z) , де $G = (V, E)$ — орграф, $z: E \rightarrow D^+$ — пропускна функція, а D^+ — множина дійсних невід'ємних чисел.

Напівстепенем виходу $In(v)$ вершини v називається сума пропускних здатностей дуг виду (v, z) . Аналогічно означається і напівстепінь входу $On(v)$. Наприклад, на мережі, зображеній на рис. 10.5.4, $In(v) = 8$, $On(x) = 10$. Очевидно, що аналог орлеми про рукостискання набуває такого вигляду: сума напівстепенів виходу всіх вершин дорівнює сумі їх напівстепенів входу. Надалі, якщо не обумовлено протилежне, будемо вважати, що орграф G має рівно одне джерело і один стік. Це обмеження несуттєве, оскільки загальний випадок легко звести до цього окремого випадку.

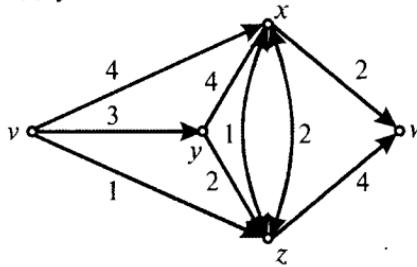


Рис. 10.5.4

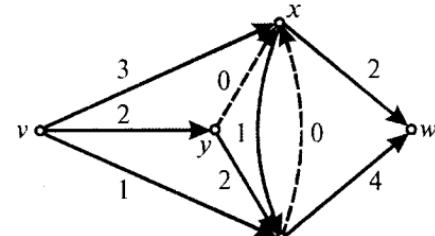


Рис. 10.5.5

Для заданої мережі $N = (G, z)$ визначимо *потік* через N як функцію h , яка ставить у відповідність кожній дузі v із G невід'ємне дійсне число $h(v)$ (що називається потоком через v) таким чином, що:

- (i) $h(v) < z(v)$ для всякої дуги v ;
- (ii) відносно мережі (G, z) напівстепінь входу і напівстепінь

виходу довільної вершини (відмінної від v і w) рівні між собою. Неформально це означає, що потік через будь-яку дугу не перевищує її пропускної здатності і що “повний потік”, який входить у будь-яку вершину (відмінну від v і w), дорівнює “повному потоку”, який виходить з неї. **Нульовим потоком** називається потік, що дорівнює нулю через будь-яку дугу даної мережі. Дуга v називається **насиченою**, якщо $z(v) = h(v)$, і ненасиченою — в протилежному випадку.

На рис. 10.5.5 зображений можливий потік для мережі, яка подана на рис. 10.5.4. Насиченими дугами цього потоку є дуги (v, z) , (x, z) , (y, z) , (x, w) і (z, w) , а решта дуг — ненасичені.

З орлеми про рукостискання випливає, що сума потоків через дуги, інцидентні вершині v , дорівнює сумі потоків через дуги, інцидентні вершині w . Ця сума називається **величиною потоку**. Як випливає з розглянутих прикладів, передусім нас будуть цікавити максимальні потоки, тобто потоки, які мають найбільшу можливу величину. Неважко перевірити, наприклад, що потік, зображеній на рис. 10.5.5, є максимальним потоком через мережу з рис. 10.5.4. Слід зауважити, що в загальному випадку максимальних потоків може бути кілька, але всі їх величини повинні бути однакові.

Вивчення максимальних потоків тісно пов’язане з поняттям розрізу в орграфі, тобто з множиною A дуг орграфа G , яка має тут властивість, що всякий простий ланцюг із v в w проходить через дугу із множини A . Інакше кажучи, розрізом у мережі є не що інше як vw -розділювальна множина відповідного орграфа. Пропускною здатністю розрізу називається сума пропускних здатностей дуг, які йому належать. Розріз, що має найменшу можливу пропускну здатність, називається **мінімальним**.

Неважко побачити, що величина будь-якого потоку не перевищує пропускної здатності довільного розрізу і, значить, величина всякого максимального потоку не перевищує пропускної здатності будь-якого мінімального розрізу. Правда, відразу не зовсім очевидно, що останні два числа завжди повинні бути рівними. Це випливає з добре відомої теореми про максимальний потік і мінімальний розріз, доведеної вперше Фордом і Фалкерсоном у 1955 р.

Теорема Форда–Фалкерсона. Для всякої мережі величина максимального потоку дорівнює пропускній здатності будь-якого мінімального розрізу.

Д о в е д е н н я. Припустимо спочатку, що пропускна здатність довільної дуги є цілим числом. Тоді мережу можна розглядати як орграф G' , в якому пропускна здатність вимірюється числом дуг, що з’єднують різні вершини. Отже, величина макси-

мального потоку в орграфі G' відповідає повному числу простих ланцюгів, які не перетинаються по ребрах і з'єднують вершини u і w , а пропускна здатність мінімального розрізу — мінімальному числу дуг в vw -розділювальній множині. Застосовуючи тепер теорему 10.5.2, одержимо потрібний результат.

Для переносу цього результату на раціональні числа необхідно помножити всі пропускні здатності на відповідне ціле число d , наприклад, на найменше спільне кратне всіх знаменників, щоб одержати цілі числа. Після цього маємо попередній випадок, і потрібний результат випливає після ділення величини максимального потоку і пропускної здатності мінімального розрізу на число d .

Аналогічно діємо у випадку, коли деякі з пропускних здатностей виражаються ірраціональними числами. У цьому випадку теорема доводиться з використанням апроксимації цих чисел раціональними дробами з довільною наперед заданою точністю. При цьому відповідні раціональні числа можна підібрати так, щоб різницю між величиною будь-якого максимального потоку і пропускною здатністю будь-якого мінімального розрізу можна зробити як завгодно малою.

Теорема доведена.

Приклад 10.5.2

Для заданої мережі (рис. 10.5.6—10.5.9) маємо, наприклад, $In(v) = 5$, $On(x) = 4$.

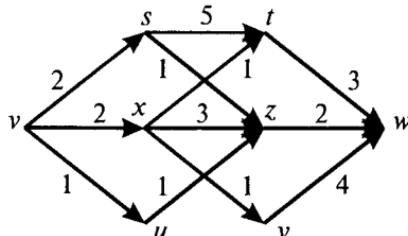


Рис. 10.5.6

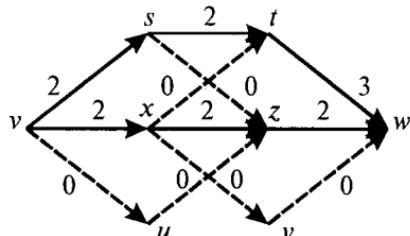


Рис. 10.5.7

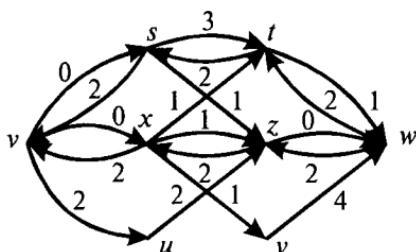


Рис. 10.5.8

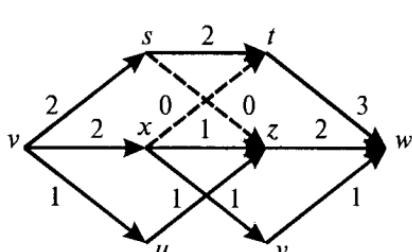


Рис. 10.5.9. ▲

Теорема Форда–Фалкерсона дає можливість перевіряти, чи є максимальним даний потік для досить простих мереж. Зрозуміло, що на практиці доводиться мати справу з величими і складними мережами, і в загальному випадку дуже важко знайти максимальний потік простим підбором. Для цього опишемо один з алгоритмів знаходження максимального потоку в довільній мережі, пропускні здатності дуг якої виражаються цілыми числами.

Нехай задана мережа $N = (G, h)$. Алгоритм знаходження максимального потоку через мінімальний розріз виконується за три кроки.

Алгоритм МПМР (G, h)

початок.

1. Насамперед підберемо нульовий потік z (якщо він існує). Слід зауважити, що чим більша величина вибраного нами початкового потоку, тим простішими будуть подальші кроки. Наприклад, якщо N — мережа, зображена на рис. 10.5.6, то підходящим потоком буде потік, зображений на рис. 10.5.7.
2. Виходячи з N , будуємо нову мережу N' шляхом зміни напрямку потоку z на протилежний. Точніше, всяка дуга e , для якої $z(e) = 0$, залишається в N' разом із своєю початковою пропускною здатністю, а всяка дуга e , для якої $z(e) = 0$, замінюється дугою e з пропускною здатністю $h(e) - z(e)$ і протилежно направленою дугою з пропускною здатністю $z(e)$. Мережа N' в нашому прикладі показана на рис. 10.5.8. Зауважимо, що вершина v уже не є джерелом, а вершина w — стоком.
3. Якщо в мережі N' ми зможемо знайти ненульовий потік із v в w , то його можна додати до початкового потоку z і одержати в N новий потік z' більшої величини. Тепер повторюється крок 2, і використовується при побудові мережі N' новий потік z' . Повторюючи цю процедуру, ми зрештою прийдемо до мережі N' , яка не має ненульових потоків. Наприклад, на рис. 10.5.8 існує ненульовий потік, в якому потоки через дуги (v, u) , (u, z) , (z, x) , (x, y) і (y, w) дорівнюють одиниці, а потоки через інші дуги — нуль. Додаючи цей потік до потоку, зображеному на рис. 10.5.6, одержимо потік, зображений на рис. 10.5.9. Повторюючи крок 2, легко переконатися, що таким чином формується максимальний потік.

кінець.

На закінчення даного розділу зауважимо, що для детальнішого ознайомлення з проблемами потоків у мережах слід звернути-ся до монографії Форда і Фалкерсона [71].

Задачі і вправи

1. Проведіть детальний покроковий розгляд роботи наведеного вище алгоритму побудови максимального потоку в мережі, яка зображена на рис 10.5.6.

2. Знайдіть максимальний потік для мережі, зображені на рис. 10.5.10.

3. Перевірте роботу даного алгоритму на прикладі мережі з раціональними пропускними здатностями.

4. Доведіть, що потік, який є результатом роботи алгоритму МПМР, буде максимальним.

5. Спроектуйте алгоритм побудови максимального потоку для мереж, графи яких двочастинні.

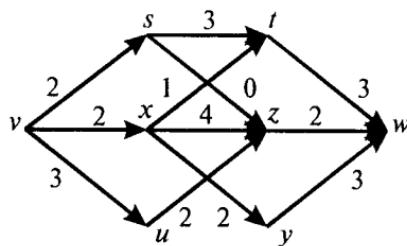


Рис. 10.5.10

МЕТОДИ ПОШУКУ ДОВЕДЕННЯ ТЕОРЕМ У ЛОГІЦІ ПРЕДИКАТІВ

Основним застосуванням логіки предикатів у комп’ютерних інформаційних технологіях є автоматизація пошуку доведень теорем. Ця проблема досить актуальна при розв’язуванні задач штучного інтелекту, задач, які виникають у базах даних і знань, при вирішенні питань представлення знань і роботи з ними.

З попередніх розділів, де розглядалися основні поняття математичної логіки, видно, що проблема доведення теорем у логіці предикатів першого порядку суттєво відрізняється від аналогічної проблеми в логіці висловлювань. Це пояснюється тим, що в логіці висловлювань існує лише скінченне число інтерпретацій, тоді як у логіці предикатів інтерпретацій даної формули може існувати нескінченно багато. Математична логіка як наука виникла в зв’язку з проблемою пошуку універсальної процедури доведення математичних тверджень. Одним із перших цією проблемою займався ще Лейбніц (1646—1716 рр.). Потім аналогічну спробу робив Пеано (кінець 19—початок 20 ст.) і Д. Гільберт зі своїми учнями. Цей пошук вівся доти, доки А. Черч і А.М. Тьюрінг незалежно один від одного не довели, що не існує алгоритму, який перевіряє тотожну істинність формул логіки предикатів першого порядку [32, 75].

Але, незважаючи на це, існують алгоритми пошуку доведення теорем, які можуть підтверджувати, що формула загальнозначима, коли вона дійсно є такою. Правда, якщо формула незагальнозначима, то ці алгоритми, взагалі кажучи, свою роботу можуть не завершити. Отже, в зв’язку з результатом Черча—Тьюрінга це найкраще, на що можна сподіватися.

Важливий внесок у методи автоматизації пошуку доведення теорем внес Ж. Ербран (1930 р.). Він запропонував алгоритм побудови інтерпретації, яка спростовує дану формулу (або її запечення).

Одним із основних сучасних методів пошуку доведення є метод резолюцій, запропонований А. Робінсоном у 1969 р. Процедура пошуку доведення, яка ґрунтуються на методі резолюцій, набагато ефективніша, ніж будь-яка з відомих процедур. У зв'язку з цим метод резолюцій детально вивчається і активно застосовується в сучасних методах пошуку доведення. Внаслідок такої уваги до методу резолюцій з'явилася багато модифікацій цього методу і стратегій його застосування, які підвищують ефективність методу. До них слід віднести такі стратегії, як семантична резолюція, лок-резолюція, лінійна резолюція [75] тощо. Перейдемо до розгляду методів Ербрана і резолюцій.

§ 11.1. МЕТОД ЕРБРАНА

1. ОСНОВНІ ОЗНАЧЕННЯ

За означенням множина диз'юнктів суперечна тоді і тільки тоді, коли вона хибна при будь-якій інтерпретації на всіх областях інтерпретації. Оскільки неможливо розглянути всі області інтерпретації, то було б корисно і зручно, якби існувала така область H , на якій множина диз'юнктів S суперечна тоді і тільки тоді, коли вона хибна при всіх інтерпретаціях на цій області. Виявляється, що така область існує, і її називають *ербранівським універсумом* множини S .

Означення 11.1.1. Нехай S — множина диз'юнктів і H_0 — множина констант, які зустрічаються в S . Якщо жодна константа не зустрічається в S , то H_0 складається з однієї константи, наприклад $H_0 = \{a\}$. Для $i = 1, 2, \dots$ множина H_{i+1} являє собою об'єднання H_i і множини всіх термів, що мають вигляд $f(t_1, t_2, \dots, t_n)$ (при всіх n) для всіх функцій f арності n , що зустрічаються в S , де $t_j \in H_i$ ($j = 1, 2, \dots, n$). Тоді кожна множина H_i називається множиною констант i -го рівня для S , і H_∞ називається *ербранівським універсумом* S .

Приклад 11.1.1

1. Нехай $S = \{P(f(x)), a, g(y), b\}$. Тоді

$$H_0 = \{a, b\},$$

$$H_1 = \{a, b, f(a), f(b), g(a), g(b)\},$$

$$H_2 = \{a, b, f(a), f(b), g(a), g(b), f(f(a)), f(f(b)), f(g(a)), f(g(b)), g(f(a)), g(f(b)), g(g(a)), g(g(b)))\}.$$

2. Нехай $S = \{P(a), \neg P(x) \vee P(f(x))\}$. Тоді

$$H_0 = \{a\},$$

$$H_1 = \{a, f(a)\},$$

$$H_2 = \{a, f(a), f(f(a))\}.$$

$$\dots\dots\dots$$

$$H_\infty = \{a, f(a), f(f(a)), f(f(f(a))), f(f(f(f(a))))\dots\}.$$

3. Нехай $S = \{P(x) \vee R(x), D(z), T(y) \vee \neg U(y)\}$. Тоді

$$H_0 = \{a\}, H_1 = H_2 = \dots = H_\infty.$$

Нижче виразом вважається не тільки терм, а й множина термів, множина атомів, літера, диз'юнкт, множина диз'юнктів. Вираз, який не містить змінних, називається **основним виразом**. Так що можна говорити про основний терм, основний атом, основну літеру, основний диз'юнкт, якщо є необхідність підкреслити, що у відповідному виразі немає змінних. Підвиразом виразу E називається вираз, який зустрічається в E .

Означення 11.1.2. Нехай S — множина диз'юнктів. Тоді множина основних атомів $A^n(t_1, \dots, t_n)$ для всіх n -арних предикатів A^n , що зустрічаються в S , де t_1, \dots, t_n — елементи ерланівського універсуму множини S , називається **множиною атомів** S або **ерланівським базисом** S .

Означення 11.1.3. Основним прикладом диз'юнкту C із множини диз'юнктів S називається диз'юнкт, одержаний внаслідок заміни змінних в C членами ерланівського універсуму S .

Приклад 11.1.2

Нехай $S = \{P(x), R(f(y)) \vee Q(y)\}$ і $C = P(x)$ — диз'юнкт в S .
Тоді

$$H = \{a, f(a), f(f(a)), f(f(f(a))) \dots\} \quad (11.1.1)$$

— ерланівський універсум множини S ;

$$A = \{P(a), Q(a), R(a), P(f(a)), Q(f(a)), R(f(a)), \dots\} \quad (11.1.2)$$

— ерланівський базис множини S і $P(a); P(f(f(a)))$ — основні приклади диз'юнкта C . *

Означення 11.1.4. Нехай S — множина диз'юнктів, H — ерланівський універсум множини S і h — деяка інтерпретація S над H . Інтерпретація h називається **H -інтерпретацією** множини S , якщо вона задовільняє такі умови:

(i) $h(a) = a$ для будь-якої константи a із H ;

(ii) якщо f — n -арний функціональний символ і b_1, b_2, \dots, b_n — елементи із H , то через f при інтерпретації h позначається функ-

ція, яка ставить у відповідність елементу $(b_1, b_2, \dots, b_n) \in H^n$ елемент $f(b_1, b_2, \dots, b_n) \in H$.

При цьому не виникає ніяких обмежень при визначенні значення будь-якого n -арного предикатного символу із S . Нехай $A = \{A_1, A_2, \dots, A_n, \dots\}$ — ербранівський базис множини S . Тоді H -інтерпретацію h зручно задавати у вигляді

$$h = \{m_1, m_2, \dots, m_n, \dots\},$$

де m_j — це або A_j , або $\neg A_j$ для $j = 1, 2, \dots, n, \dots$, причому, якщо $m_j = A_j$, то атому A_j присвоєно значення 1, і якщо $m_j = \neg A_j$, — то значення 0.

Приклад 11.1.3

Розглянемо множину $S = \{P(x), R(f(y)) \vee Q(y)\}$. Ербранівський універсум H для S і його ербранівський базис мають вигляд (11.1.1) і (11.1.2). H -інтерпретаціями множини S будуть, наприклад, такі інтерпретації:

$$\begin{aligned} h_1 &= \{P(a), Q(a), R(a), P(f(a)), Q(f(a)), R(f(a)), \dots\}; \\ h_2 &= \{\neg P(a), \neg Q(a), \neg R(a), \neg P(f(a)), \neg Q(f(a)), \neg R(f(a)), \dots\}; \\ h_3 &= \{P(a), \neg Q(a), R(a), P(f(a)), \neg Q(f(a)), R(f(a)), \dots\}; \\ h_4 &= \{P(a), Q(a), \neg R(a), P(f(a)), Q(f(a)), \neg R(f(a)), \dots\}. \end{aligned}$$

Нехай задана деяка область D і інтерпретація z множини діз'юнктів S над D . Виникає питання: яким чином пов'язані між собою H -інтерпретація і інтерпретація z множини діз'юнктів S ?

Нехай $D = \{1, 2\}$ і $S = \{P(x), Q(y, f(y, a))\}$. Тоді можлива така інтерпретація z над D :

$$\begin{aligned} a &= 2, f(1, 1) = 1, f(1, 2) = 2, f(2, 1) = 2, f(2, 2) = 1, \dots; \\ P(1) &= 1, P(2) = 0, Q(1, 1) = 0, Q(1, 2) = 1, Q(2, 1) = 0, \\ Q(2, 2) &= 1, \dots. \end{aligned}$$

Для такої інтерпретації z можна визначити H -інтерпретацію h , яка відповідає z . Це виконується таким чином. Спочатку побудуємо ербранівський базис S :

$$\begin{aligned} A &= \{P(a), Q(a, a), P(f(a, a)), Q(a, f(a, a)), \\ &\quad Q(f(a, a), a), Q(f(a, a), f(a, a)), \dots\}. \end{aligned}$$

Тепер, користуючись інтерпретацією z , знайдемо значення кожного члена ербранівського базису A :

$$\begin{aligned} P(a) &= P(2) = 0, \\ Q(a, a) &= Q(2, 2) = 1, \\ P(f(a, a)) &= P(f(2, 2)) = P(1) = 1, \\ Q(a, f(a, a)) &= Q(2, f(2, 2)) = Q(2, 1) = 0, \\ Q(f(a, a), a) &= Q(f(2, 2), 2) = Q(1, 2) = 1, \end{aligned}$$

$$Q(f(a, a), f(a, a)) = Q(f(2, 2), f(2, 2)) = Q(1, 1) = 0.$$

Отже, H -інтерпретація h , яка відповідає інтерпретації z , має вигляд

$$h = \{\neg P(a), Q(a, a), P(f(a, a)), \neg Q(a, f(a, a)), \\ Q(f(a, a), a), \neg Q(f(a, a), f(a, a), \dots\}.$$

Якщо в множині S відсутні константи, то елемент a ербранівського універсуму може мати довільний образ в області D . Якщо ж D складається більше ніж з одного елемента, то існують більше однієї інтерпретації, які відповідають інтерпретації z .

Означення 11.1.5. Нехай z — інтерпретація на області D . H -інтерпретацією h , що відповідає z , називається інтерпретація, для якої виконуються такі умови.

Нехай b_1, b_2, \dots, b_n — елементи ербранівського універсуму H , і кожний елемент b_i відображається в деякий елемент d_i з області D . Тоді якщо $P(d_1, d_2, \dots, d_n)$ одержує в інтерпретації z значення 1(0), то $P(b_1, b_2, \dots, b_n)$ також одержує значення 1 (відповідно 0) при інтерпретації h .

Лема 11.1.1. Якщо інтерпретація z на деякій області D задоволяє множину диз'юнктів S , то довільна H -інтерпретація h , що відповідає інтерпретації z , також задоволяє множину S .

Д о в е д е н н я. Нехай інтерпретація z задовольняє множину диз'юнктів S на області D . Це означає, що довільна формула із S приймає значення 1 на D при інтерпретації z . Але тоді за побудовою інтерпретації h , що відповідає z , кожна з формул множини S теж приймає значення 1 при інтерпретації h , тобто задоволяє множину S .

Теорема 11.1.1. Множина диз'юнктів S суперечна тоді і тільки тоді, коли S хибна при всіх H -інтерпретаціях множини S .

Д о в е д е н н я. В один бік теорема очевидна, оскільки за означенням множина S хибна тоді і тільки тоді, коли S хибна при будь-якій інтерпретації на довільній області.

Щоб довести теорему в другий бік, припустимо, що S хибна при всіх H -інтерпретаціях S . Припустимо також, що S приймає значення 1 при деякій інтерпретації z на деякій області D і h — H -інтерпретація S , що відповідає z . Тоді за лемою 11.1.1 S приймає значення 1 при інтерпретації h . А це суперечить тому, що S хибна при всіх H -інтерпретаціях множини S .

Теорема доведена.

Доведена теорема дає можливість розглядати лише інтерпретації над ербранівським універсумом (H -інтерпретації) для перевірки, чи є множина диз'юнктів S хибною. Отже, надалі, коли мова йтиме про інтерпретацію, будемо мати на увазі H -інтерпретацію.

2. СЕМАНТИЧНІ ДЕРЕВА

Нагадаємо, якщо A — атом, то дві літери A і $\neg A$ називаються *контрарними*. Диз'юнкт є тавтологією, якщо він має контрарну пару.

Означення 11.1.6. Нехай S — множина диз'юнктів і A — її ербранівський базис. Семантичним деревом множини S називається дерево T , ребрам якого поставлена у відповідність скінченна множина атомів або заперечень атомів із A так, що:

1) ізожної вершини N дерева T виходить тільки скінченне число ребер L_1, L_2, \dots, L_n ; нехай Q_i — кон'юнкція всіх літер, які поставлені у відповідність ребру L_i , $i = 1, 2, \dots, n$; тоді $Q_1 \vee Q_2 \vee \dots \vee Q_n$ — загальнозначима пропозиційна формула;

2) нехай дляожної вершини N дерева T $h(N)$ являє собою об'єднання всіх множин, які поставлені у відповідність ребрам, що складають шлях із початкової вершини T у вершину N . Тоді $h(N)$ не має контрапарних пар.

Означення 11.1.7. Нехай $A = \{A_1, A_2, \dots, A_n, \dots\}$ — ербранівський базис множини S . Семантичне дерево T для S називається *повним* тоді і тільки тоді, коли для кожного i ($i = 1, 2, \dots$) і кожної вершини-листка N дерева T $h(H)$ включає або A_i , або $\neg A_i$.

Приклади 11.1.4

1. Нехай $S = \{P, Q, R\}$ — ербранівський базис множини S . Тоді кожне з двох дерев, зображених на рис. 11.1.1, являє собою повне семантичне дерево множини S .

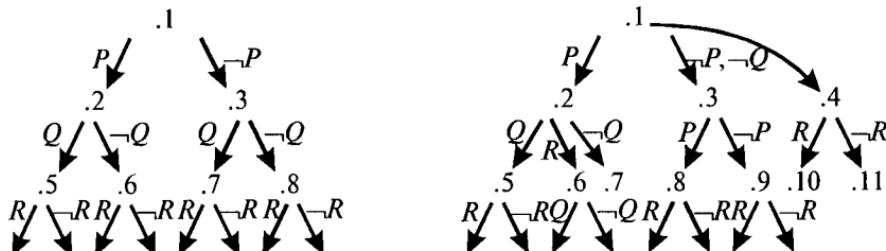


Рис. 11.1.1

2. Розглянемо $S = \{P(x), Q(f(x))\}$ і його ербранівський базис $A = \{P(a), Q(f(a)), P(f(a)), Q(f(f(a))), \dots\}$.

Семантичне дерево для S наведено на рис. 11.1.2.

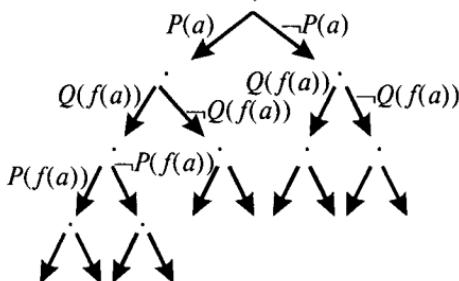


Рис. 11.1.2

Зазначимо, що для кожної вершини N в семантичному дереві для S $h(N)$ є підмножиною відповідної інтерпретації для S . Отже, $h(N)$ логічно називати частковою інтерпретацією S . Крім того, якщо ербранівський базис множини S нескінчений, то і семантичне дерево T для S теж буде нескінченим. Неважко помітити, що повне семантичне дерево відповідає повному переліку всіх можливих інтерпретацій множини S . Якщо S суперечна, то S не може бути істинною в жодній із цих інтерпретацій. Значить, ми можемо зупинити ріст семантичного дерева T з вершини N , якщо $h(N)$ спростовує S .

Наведені зауваження дають можливість ввести такі означення.

Означення 11.1.8. Вершина N семантичного дерева T для множини диз'юнктів S називається **вершиною-спростуванням**, якщо $h(N)$ спростовує деякий основний приклад диз'юнкта із S , і для всякої вершини N' , що зустрічається на шляху з початкової вершини у вершину N , $h(N')$ не є спростуванням жодного основного прикладу диз'юнкта з S .

Означення 11.1.9. Семантичне дерево T множини диз'юнктів S називається закритим тоді і тільки тоді, коли кінцевою вершиною кожного шляху з початкової вершини дерева T є вершина-спростування.

Означення 11.1.10. Вершина N закритого семантичного дерева T називається **вершиною-виведенням**, якщо кожний син вершини N є вершиною-спростуванням.

3. ТЕОРЕМА ЕРБРАНА

Теорема Ербрана, яка формулюється і доводиться нижче, є однією з найважливіших теорем математичної логіки, оскільки вона дає алгоритмічну основу для перевірки суперечності множини диз'юнктів. Розглянемо два варіанти цієї теореми.

Теорема Ербрана (варіант 1). *Множина диз'юнктів S суперечна тоді і тільки тоді, коли всякому повному семантичному дереву множини S відповідає скінченне закрите семантичне дерево.*

Доведення. Припустимо, що S суперечна і нехай T — семантичне дерево для S . Нехай для кожного шляху l з початкової вершини в дереві T h_l означає множину всіх літер, які поставлені у відповідність ребрам шляху l . Тоді h_l є інтерпретацією множини S . Оскільки S суперечна, то h_l повинна спростовувати основний приклад C' диз'юнкта C із S . Оскільки C' скінченний, то на шляху l повинна існувати вершина-спростування N_l , і l теж має бути скінченним. Крім того, оскільки кожний шлях виду l в дереві T має вершину-спростування, то існує закрите семантичне дерево T' для множини S , і оскільки кожна вершина в T' має скінченне число суміжних з нею вершин, то за лемою Кьюніга T' — скінченне дерево.

Навпаки, якщо для всякого повного семантичного дерева T множини S існує скінченне закрите семантичне дерево, то всякий шлях у T має вершину-спростування. А це значить, що всяка інтерпретація спростовує множину S . Отже, S — суперечна множина диз'юнктів.

Теорема доведена.

Теорема Ербрана (варіант 2). *Множина диз'юнктів S суперечна тоді і тільки тоді, коли існує скінченна суперечна множина S' основних прикладів диз'юнктів S .*

Доведення. Припустимо, що S — суперечна множина і T — повне семантичне дерево для S . Тоді за теоремою Ербрана (варіант 1) існує скінченне закрите семантичне дерево T' , яке відповідає T . Нехай S' — множина всіх основних прикладів диз'юнктів, які спростовуються у всіх вершинах-спростуваннях дерева T' . Множина S' скінчена, оскільки в T' скінченне число вершин-спростувань. Оскільки S' хибна при будь-якій інтерпретації, то S' суперечна.

Нехай тепер існує скінченна суперечна множина S' основних прикладів диз'юнктів із S . Оскільки всяка інтерпретація h для S включає інтерпретацію h' множини S' і h' спростовує S' , то h теж повинна спростовувати S' . Оскільки S' спростовується в кожній інтерпретації h' , то і S' спростовується в кожній інтерпретації h множини S . Отже, S спростовується в кожній інтерпретації множини S' , і, значить, S — суперечна.

Теорема доведена.

На варіанті 2 теореми Ербрана ґрунтуються процедура спростування (метод Ербрана). Це означає, що коли необхідно довести суперечність множини диз'юнктів S і ми маємо алгоритм,

який породжує множини $S'_1, S'_2, \dots, S'_n, \dots$ основних прикладів диз'юнктів із S і перевіряє їх суперечність, то цей алгоритм, як гарантує теорема Ербрана, дасть нам такий індекс N , що S_N суперечна. Слід зауважити, що описаний метод малоефективний. Ефективнішим щодо застосувань є метод резолюції.

§ 11.2. МЕТОД РЕЗОЛЮЦІЙ

1. КОНТРАРНІ ПАРИ І ПІДСТАНОВКИ

З розглянутого вище для логіки висловлювань випливає, що суттєвим для методу резолюції є пошук контрапарних пар літер для заданої множини диз'юнктів. У логіці предикатів цей пошук ускладнюється, оскільки з'являються змінні. Пояснимо це на прикладі.

Нехай

$$\begin{aligned} C_1 &= P(y) \vee R(y), \\ C_2 &= \neg P(f(x)) \vee Q(x). \end{aligned}$$

Не існує жодної літери в C_1 , контрапарної якій-небудь літері в C_2 . Але коли підставити терм $f(a)$ замість y в диз'юнкті C_1 і a — замість x в диз'юнкті C_2 , то одержимо

$$\begin{aligned} C'_1 &= P(f(a)) \vee R(f(a)), \\ C'_2 &= \neg P(f(a)) \vee Q(a). \end{aligned}$$

В цих диз'юнктах літери $P(f(a))$ і $\neg P(f(a))$ вже контрапарні. Отже, з C'_1 і C'_2 можна одержати резольвенту

$$C'_3 = Q(a) \vee R(f(a)).$$

У загальному випадку, підставивши вираз $f(x)$ замість y в C_1 , одержимо

$$C'_1 = P(f(x)) \vee R(f(x)).$$

Диз'юнкт C'_1 дає можливість одержати резольвенту

$$C_3 = R(f(x)) \vee Q(x).$$

Підставивши деякі терми замість x в цьому диз'юнкті, можна одержати нові диз'юнкти. Диз'юнкт C_3 — найзагальніший диз'юнкт, оскільки всі інші диз'юнкти, які можна породити за допомогою правила резолюції з C_1 і C_2 , — це конкретизації диз'юнкта C_3 (такі конкретизації називаються прикладами диз'юнкта C_3).

Отже, для одержання резольвент із диз'юнктів необхідно виконувати підстановки відповідних термів замість змінних.

Нехай X — алфавіт змінних і $T(\Omega, X)$ — алгебра термів над X .
Підстановкою називається відображення σ , що має вигляд

$$\sigma: X \rightarrow T(\Omega, X).$$

Підстановка σ називається **скінченою**, якщо $\sigma(x) \neq x$ лише для скінченого числа елементів x із X . Підстановка, яка не має елементів, називається **тотожною** і позначається ε .

Приклад 11.2.1

Множини пар $\{(x, f(z)), (z, y)\}$ і $\{(x, a), (y, g(y)), (z, f(g(b)))\}$ є підстановками, де

$$\begin{aligned}\sigma(x) &= f(z), \sigma(z) = y; \\ \sigma(x) &= a, \sigma(y) = g(y), \sigma(z) = f(g(b)).\end{aligned} \quad \blacktriangleleft$$

Підстановку σ часто позначають $\{x/t_1, y/t_2, \dots, z/t_n, \dots\}$ або $\{x \rightarrow t_1, y \rightarrow t_2, \dots, z \rightarrow t_n, \dots\}$.

Нехай $s = \{x_1/t_1, x_2/t_2, \dots, x_n/t_n\}$ — скінчена підстановка і $t \in T(\Omega, X)$. Тоді терм t' , одержаний з терма t одночасною заміною всіх входжень змінної x_i в термі t термами t_i ($1 \leq i \leq n$), позначається $\sigma(t)$ і називається **прикладом терма** t .

Приклад 11.2.2

Нехай $\sigma = \{x/a, y/f(c), z/b\}$ і $t = \omega(x, y, z)$. Тоді $t' = \sigma(t) = \omega(a, f(c), b)$ — приклад терма t . \blacktriangleleft

Нехай $\sigma = \{x_1/t_1, x_2/t_2, \dots, x_n/t_n\}$, $v = \{y_1/t'_1, y_2/t'_2, \dots, y_m/t'_m\}$ — дві підстановки.

Добутком підстановок σ і v називається підстановка, одержана з множини

$$\{x_1/v(t_1), \dots, x_n/v(t_n), y_1/t'_1, \dots, y_m/t'_m\}$$

виолученням всіх елементів, що мають вигляд $x_j/v(t_j)$, для яких $v(t_j) = x_j$, і всіх елементів, що мають вигляд y_i/t'_i і таких, що $y_i \in \{x_1, x_2, \dots, x_n\}$.

Приклад 11.2.3

Нехай

$$\begin{aligned}\sigma &= \{x_1/t_1, x_2/t_2\} = \{x/f(y), y/z\}, \\ v &= \{y_1/t'_1, y_2/t'_2, y_3/t'_3\} = \{x/a, y/b, z/y\}.\end{aligned}$$

Тоді

$$\{x/f(y), y/z, x/a, y/b, z/y\}.$$

Але оскільки $v(t_2) = y$ (тобто y/y), то ця пара вилучається із

одержаної множини. Далі, підстановки x/a і y/b також вилучаються з цієї множини, оскільки x і y належать множині змінних $\{x_1, x_2\}$ першої підстановки. Таким чином, остаточно маємо

$$\sigma v = \{x/f(b), z/y\}. \blacksquare$$

Слід зазначити, що операція добутку підстановок асоціативна, а totожна підстановка ϵ відіграє роль одиниці для цієї операції.

Означення 11.2.1. Підстановка σ називається уніфікатором для множини термів $\{t_1, t_2, \dots, t_k\}$ тоді і тільки тоді, коли $\sigma(t_1) = \sigma(t_2) = \dots = \sigma(t_k)$. Говорять, що множина термів $\{t_1, t_2, \dots, t_k\}$ уніфікується, якщо для неї існує уніфікатор.

Уніфікатор σ множини термів $\{t_1, t_2, \dots, t_k\}$ називається *найбільшим загальним уніфікатором* (НЗУ) тоді і тільки тоді, коли для всякого уніфікатора v цієї множини існує така підстановка λ , що $v = \lambda\sigma$.

2. АЛГОРИТМИ УНІФІКАЦІЇ

Розглянемо тепер проблему уніфікації детальніше. Розширимо спочатку поняття уніфікатора таким чином: нехай $t = (t_1, t_2, \dots, t_n)$, $t' = (t'_1, t'_2, \dots, t'_n)$ — вектори термів. Тоді їх уніфікатором назовемо підстановку σ , таку, що

$$\begin{aligned}\sigma(t_1) &= \sigma(t'_1), \\ \sigma(t_2) &= \sigma(t'_2), \\ &\dots \\ \sigma(t_n) &= \sigma(t'_n).\end{aligned}$$

Нехай $\text{Unify}(t_1, t_2)$ означає функцію уніфікації термів t_1 і t_2 , тобто функцію побудови НЗУ цих термів (або векторів із термів). Введемо також символ Λ для ніде не визначеної підстановки. Будемо вважати, що для всякої підстановки σ справедливі рівності

$$\sigma * \Lambda = \Lambda * \sigma = \Lambda \quad \text{i} \quad \sigma * \epsilon = \epsilon * \sigma = \sigma.$$

Тепер правило побудови НЗУ можна задати таким індуктивним означенням:

$$(a) \text{Unify}(t_1, t_2) = \text{Unify}(t_2, t_1);$$

$$(b) \text{Unify}(t, z) =$$

$$= \begin{cases} \Lambda, & \text{коли терм залежить від змінної } z \text{ i } t \neq z, \\ \epsilon, & \text{коли } t = z, \\ \{(z/t)\} - \text{ в інших випадках}; \end{cases}$$

$$(в) \text{Unify}((t_1, \dots, t_n), (t'_1, \dots, t'_k)) = \\ = \begin{cases} \Lambda, & \text{коли } k \neq n, \\ \sigma * \text{Unify}((\sigma(t_2), \dots, \sigma(t_n)), (\sigma(t'_2), \dots, \sigma(t'_n))), \end{cases}$$

де $\sigma = \text{Unify}(t_1, t'_1)$;

$$(г) \text{Unify}(\omega(t_1, \dots, t_n), \omega'(t'_1, \dots, t'_n)) = \\ = \begin{cases} \Lambda, & \text{коли } \omega \neq \omega', \\ \text{Unify}((t_1, \dots, t_n), (t'_1, \dots, t'_n)) & - \text{ в інших випадках.} \end{cases}$$

Користуючись даним означенням, неважко показати, що підстановка, яка будується за такими правилами, буде НЗУ термів t_1 і t_2 . Іншими словами, має місце така теорема.

Теорема 11.2.1. $\text{Unify}(t_1, t_2)$ являє собою НЗУ термів t_1 і t_2 в абсолютно вільній алгебрі.

Доведення пропонується як вправа.

Приклад 11.2.4

Знайти НЗУ таких термів:

- a) $((x + y) * (u / v)) - 1 + 2 i ((z / w) - 1) + r;$
 б) $((x + y) * z) - 1 + 2 i ((z / w) - 1) + 3.$

Маємо:

$$\begin{aligned} a) \text{Unify}(&+(-(*(+x, y), /(u, v)), 1), 2; +(-(/(z, w), 1), r)) = \\ = \text{Unify}(&-(*(+x, y), /(u, v)), 1); -(/(z, w), 1), r) * \text{Unify}(2, r) = \\ = \text{Unify}(&(*(+x, y); /(u, v)); /(z, w)) * \text{Unify}(1, 1) * \text{Unify}(2, r) = \\ = \text{Unify}(&(+x, y); z) * \text{Unify}(/(u, v); w) * \text{Unify}(2, r) = \\ = \{z/(x+y), w/(u/v), r/2\}; \\ b) \text{Unify}(&+(-(*(+x, y), /(u, v)), 1), 2; +(-(/(z, w), 1), 3)) = \\ = \text{Unify}(&-(*(+x, y), /(u, v)), 1); -(/(z, w), 1)) * \text{Unify}(2, 3) = \\ = \text{Unify}(&-(*(+x, y), /(u, v)), 1); -(/(z, w), 1)) * \Lambda = \Lambda. \end{aligned}$$

Алгоритм Робінсона. На даному означенні НЗУ ґрунтуюється алгоритм уніфікації, запропонований А. Робінсоном [75]. Для роботи цього алгоритму початкові терми зображаються розміченими деревами, і тому цей алгоритм інколи називають *Robinson's tree algorithm (RT)*.

RT: Unify(t_1, t_2)

Вхід: $t_1, t_2 \in T(\Omega, X)$ — пара термів.

Вихід: (*bool*, σ), де *bool* = 1 тоді і тільки тоді, коли t_1 і t_2 уніфікуються і $\sigma = \text{НЗУ}(t_1, t_2)$.

початок.

якщо (один із термів x є змінною, а другий t — термом) **то**

якщо $x = t$ **то** (*bool* = 1; $\sigma = \epsilon$) **інакше**

якщо t залежить від x то $\text{bool} = 0$ інакше
 ($\text{bool} = 1;$
 $\sigma = \{x/t\}$)
 кя;
 кя;
 інакше (нехай $t = \omega(x_1, \dots, x_m)$, $t' = \omega'(y_1, \dots, y_m)$)
 якщо $\omega \neq \omega'$ то $\text{bool} = 0$ інакше
 ($k = 0;$
 $\text{bool} = 1;$
 $\sigma = \varepsilon;$
 поки $((k < m) \& \text{bool})$ виконувати
 $k = k + 1;$
 $(\text{bool}, \sigma_1) = \text{Unify}(\sigma(x_k), \sigma(y_k));$
 якщо bool то $\sigma = \sigma_1 * \sigma$; кя;
 кш;
 кя;
 кя;
 кінець.

Модифікований алгоритм Робінсона. Алгоритм Робінсона, як неважко помітити, має експоненціальну оцінку часової складності. Одна з причин повільної роботи даного алгоритму — задання термів деревами, при якому часто доводиться обчислювати значення підтермів при побудові НЗУ. Цього можна уникнути, якщо добудовувати підтерми, що уніфікуються, в міру того, як вдається виразити значення одних змінних через інші. У цьому випадку для підстановки σ переход від терма $\omega(t_1, \dots, t_n)$ до значень $\sigma(t_1), \dots, \sigma(t_n)$ його підтермів t_1, \dots, t_n виконується у вигляді звичайного переходу від дерева до його піддерев. Зауважимо, що при такій домовленості про динамічну добудову термів у момент завершення роботи $\text{Unify}(t_1, t_2)$ значення термів $\sigma(t_1)$ і $\sigma(t_2)$ будуть рівні, де $\sigma = \text{Unify}(t_1, t_2)$.

Інша, більш тонка можливість прискорення роботи алгоритму уніфікації випливає безпосередньо з означення НЗУ:

$$\text{Unify}(t, t) = \varepsilon.$$

Вказана властивість має такий вигляд:

$$t_1 = t_2 \rightarrow \text{Unify}(t_1, t_2) = \varepsilon.$$

Використовуючи рекурсивний виклик *Unify* в тілі алгоритму *RT* можна замінити його умовним рекурсивним викликом:

якщо $t_1 \neq t_2$, то $\text{Unify}(t_1, t_2).$

Безпосереднє порівняння термів є досить трудомістким про-

цесом, який ускладнюється ще й тим, що внаслідок підстановок розміри термів можуть значно зростати. Таким чином, суттєвого зменшення часової складності алгоритму можна досягти при таких структурах даних, коли рівні підтерми задаються одним і тим же об'єктом. Таку властивість мають розмічені ациклічні графи з приведеними спільними підтермами.

Зауважимо, що коли $\sigma = \text{Unify}(t_1, t_2)$ і $\sigma \neq \epsilon$, то терми $\sigma(t_1)$ і $\sigma(t_2)$ можна склеїти. Отже, після кожного виходу з процедурі *Unify*, крім тих, які виробляють значення Λ , необхідно виконувати склеювання підграфів ациклічного графа. Для цього введемо до розгляду процедуру *Склейти* (G, v_1, v_2), яка виконує заміну всіх дуг у графі G , що ведуть у вершину v_1 , дугами, що ведуть у вершину v_2 . Введемо також позначення $\text{term}(v)$ для терма, який є результатом розклейовання ациклічного графа з вершиною v , і $\text{succ}(v, k)$ — для k -го сина вершини v .

В результаті модифікації алгоритм уніфікації *RT* набуває такого вигляду:

RG: Unify(v_1, v_2)

Vxid: v_1, v_2 — пара різних вершин розміченого ациклічного графа.

Buxid: (bool, σ), де $\text{bool} = 1$ тоді і тільки тоді, коли t_1 і t_2 уніфікуються і $\sigma = \text{НЗУ}(t_1, t_2)$.

початок.

якщо (один із термів u є змінною, а другий v — термом) **то**

якщо (u досяжна із v) **то** $\text{bool} = 0$ **інакше**

$(\sigma = \{\text{term}(u)/\text{term}(v)\};$

$\text{bool} = 1;$

Склейти (G, u, v)

кя;

інакше

якщо $\text{отм}(v_1) \neq \text{отм}(v_2)$ **то** $\text{bool} = 0$ **інакше**

*/** нехай m — число синів вершин v_1 і v_2 **/*

$(k = 0;$

$\text{bool} = 1;$

$\sigma = \epsilon;$

поки $((k < m) \& \text{bool}) **виконувати**$

$k = k + 1;$

$t = \text{succ}(u, k);$

$t' = \text{succ}(v, k);$

якщо $t \neq t'$ **то** $((\text{bool}, \sigma_1) = \text{Unify}(t, t');$

якщо bool **то** $\sigma = \sigma_1 * \sigma$; **кя;**

кц;

кя;

кя;
кінець.

З тексту алгоритму RG видно, що при всякому виході з процедури `Unify`, коли вона не закінчується результатом Λ ($\text{bool} == 0$), виконується склеювання, і одна з вершин ацикличного графа ізоляється. Отже, загальне число звернень до цієї процедури не перевищує N — числа вершин у графах, які представляють початкові терми t_1 і t_2 . В той же час число дій, які виконуються всередині процедури без урахування дій, що виконуються при рекурсивних зверненнях до неї, теж пропорційне числу N , тобто стільки часу потрібно для перевірки, чи є вершина v досяжною з вершини v . Таким чином, верхня оцінка алгоритму RG пропорційна числу N^2 [20].

3. МЕТОД РЕЗОЛЮЦІЙ ДЛЯ ЛОГІКИ ПРЕДИКАТИВ ПЕРШОГО ПОРЯДКУ (ЛППП)

Розв'язавши проблему уніфікації, можна розглядати метод резолюцій і для ЛППП.

Якщо дві або більше літер з однаковими символами предиката в диз'юнкті C мають НЗУ σ , то $\sigma(C)$ називається *склеюванням* C . Коли $\sigma(C)$ — одиничний диз'юнкт, то склеювання $\sigma(C)$ називається *одиничним склеюванням*.

Приклад 11.2.5

Нехай $C = P(x) \vee P(g(y)) \vee \neg R(x)$. Тоді перша і друга літери мають однакові символи предиката і НЗУ $\sigma = \{x/g(y)\}$. Значить,

$\sigma(C) = P(g(y)) \vee P(g(y)) \vee \neg R(g(y)) = P(g(y)) \vee \neg R(g(y))$
являє собою склеювання C . \blacktriangleleft

Означення 11.2.2. Нехай C_1 і C_2 — два диз'юнкти (диз'юнкти-посилки), які не мають спільних змінних, а L_1 і L_2 — дві літери в C_1 і C_2 відповідно. Якщо L_1 і $\neg L_2$ мають НЗУ σ , то диз'юнкт

$$(\sigma(C_1) \setminus \sigma(L_1)) \cup (\sigma(C_2) \setminus \sigma(L_2))$$

називається *бінарною резольвентою* C_1 і C_2 . Літери L_1 і L_2 називаються такими, що відрізуються.

Приклад 11.2.6

Нехай $C_1 = P(x) \vee R(x)$ і $C_2 = \neg P(a) \vee Q(x)$. Оскільки змінна x

входить як в C_1 , так і в C_2 , то замінимо x в C_2 на y . Тоді $C_2 = \neg P(a) \vee Q(y)$. Вибираємо літери $L_1 = P(x)$ і $L_2 = \neg P(a)$. Оскільки $\neg L_2 = P(a)$, а L_1 і L_2 мають НЗУ $\sigma = \{x/a\}$, то

$$(\sigma(C_1) \setminus \sigma(L_1)) \cup (\sigma(C_2) \setminus \sigma(L_2)) = (\{P(a), R(a)\} \setminus \{P(a)\}) \cup (\{\neg P(a), Q(y)\} \setminus \{\neg P(a)\}) = \{R(a)\} \cup \{Q(y)\} = R(a) \vee Q(y).$$

Отже, $R(a) \vee Q(y)$ — бінарна резольвента C_1 і C_2 , а $P(x)$ і $\neg P(a)$ — літери, що відрізуються. \blacktriangleleft

Означення 11.2.3. Резольвентою диз'юнктів-посилок C_1 і C_2 називається одна з таких резольвент:

- 1) бінарна резолюція $C_1 \wedge C_2$;
- 2) бінарна резолюція C_1 і склеювання C_2 ;
- 3) бінарна резолюція C_2 і склеювання C_1 ;
- 4) бінарна резолюція склеювання C_1 і склеювання C_2 .

Приклад 11.2.7

Нехай $C_1 = Q(x) \vee Q(g(y)) \vee R(f(y))$ і $C_2 = \neg Q(g(f(a))) \vee Q(b)$. Склєюванням $C_1 \in C'_1 = Q(g(y)) \vee Q(g(y)) \vee R(f(y)) = Q(g(y)) \vee \vee R(f(y))$. Бінарною резольвентою C'_1 і C_2 є $R(f(f(a))) \vee Q(b)$. Значить, $R(f(f(a))) \vee Q(b)$ являє собою резольвенту C_1 і C_2 . \blacktriangleleft

Правило резолюцій для ЛППП є правилом виведення. Воно ефективніше за такі процедури, як метод редукції, метод Девіса—Патнема та ін. [75]. Крім того, як уже зазначалося вище, метод резолюцій є повним. Перед тим як навести доведення повноти методу резолюцій, розглянемо застосування цього методу для доведення однієї теореми з елементарної геометрії.

Приклад 11.2.8

Довести, що внутрішні несуміжні кути, які утворюються при перетині діагоналей трапеції з її основами, рівні.

Для доведення цієї теореми необхідно спочатку її аксіоматизувати. Нехай $T(x, y, u, v)$ означає, що $xuyv$ — трапеція, верхня ліва вершина якої x , верхня права — y , нижня права — u і нижня ліва — v . Нехай $P(x, y, u, v)$ означає, що відрізок xu паралельний відрізку uv , і $E(x, y, z, u, v, w)$ означає, що кут xuz дорівнює куту uvw . Тоді мають місце такі аксіоми:

$A_1 : \forall x \forall y \forall u \forall v (T(x, y, u, v) \rightarrow P(x, y, u, v))$ — означення трапеції;

$A_2 : \forall x \forall y \forall u \forall v (P(x, y, u, v) \rightarrow E(x, y, v, u, v, w))$ — внутрішні несуміжні кути при перетині паралельних прямих рівні;

$A_3 : T(a, b, c, d)$ — трапеція (показана на рис. 11.2.1).

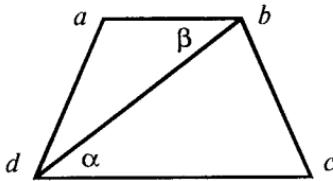


Рис. 11.2.1

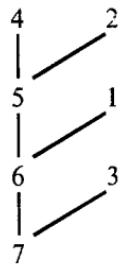


Рис. 11.2.2

З аксіом A_1, A_2, A_3 ми повинні вивести, що твердження $E(a, b, d, c, d, b)$ справедливе, тобто що

$$A_1 \& A_2 \& A_3 \rightarrow E(a, b, d, c, d, b).$$

Для того щоб скористатися методом резолюцій, тобто довести цю формулу шляхом спростування, ми заперечуємо висновок, тобто формулу $\neg E(a, b, d, c, d, b)$, і доводимо, що формула $A_1 \& A_2 \& A_3 \& \neg E(a, b, d, c, d, b)$ хибна.

Перетворимо дану формулу до стандартного вигляду:

$$S = \{\neg T(x, y, u, v) \vee P(x, y, u, v), \neg P(x, y, u, v) \vee E(x, y, v, u, v, w), \\ T(a, b, c, d), \neg E(a, b, d, c, d, b)\}.$$

Ця стандартна форма є множиною з чотирьох диз'юнктів. Тепер методом резолюцій виконаємо доведення.

- | | |
|---|----------------------|
| (1) $\neg T(x, y, u, v) \vee P(x, y, u, v)$ | — диз'юнкт 1 в S , |
| (2) $\neg P(x, y, u, v) \vee E(x, y, v, u, v, w)$ | — диз'юнкт 2 в S , |
| (3) $T(a, b, c, d)$ | — диз'юнкт 3 в S , |
| (4) $\neg E(a, b, d, c, d, b)$ | — диз'юнкт 4 в S , |
| (5) $\neg P(a, b, c, d)$ | — резольвента 2 і 4, |
| (6) $\neg T(a, b, c, d)$ | — резольвента 1 і 5, |
| (7) 0 | — резольвента 3 і 6. |

Оскільки останній виведений із S диз'юнкт пустий, то робимо висновок, що S — суперечна множина. Отже, тоді теорема має місце.

Кроки виведення легко можна представити деревом (рис. 11.2.2).

Це дерево називається деревом виведення і являє собою дерево, яке “росте вгору”. Кожній початковій вершині даного дерева приписується диз'юнкт із множини S , а кожній наступній вершині — резольвента диз'юнктів, які були приписані її предкам.

Древо виведення називається *деревом виведення диз'юнкта R*, якщо R приписаний кореню дерева виведення. Надалі “дерево

виведення” і “виведення” будуть використовуватися як синоніми. ▲

Із сказаного вище випливає, що наша мета полягає в доведенні такого твердження.

Теорема 11.2.2 (повнота методу резольюцій). *Множина диз'юнктів S суперечна тоді і тільки тоді, коли існує виведення із множини S пустого диз'юнкта 0.*

Перед тим як довести цю теорему, доведемо таке допоміжне твердження.

Лема 11.2.1. *Якщо C_1' і C_2' — приклади C_1 і C_2 відповідно, і якщо C' — резольвента C_1' і C_2' , то існує така резольвента C із C_1 і C_2 , що C' є прикладом C .*

Д о в е д е н н я. Перенумеруємо, якщо є необхідність, змінні в C_1 і C_2 так, щоб C_1 і C_2 не мали спільних змінних. Нехай L_1' і L_2' — літери, що відрізуються, і нехай

$$C' = (C_1'v \setminus L_2'v) \cup (C_2'v \setminus L_2'v),$$

де v — НЗУ для L_1' і $\neg L_2'$. Оскільки C_1' і C_2' — приклади C_1 і C_2 відповідно, то існує така підстановка v , що $C_1' = C_1v$ і $C_2' = C_2v$. Нехай $L_i^1, \dots, L_i^{j_i}$ — літери в C_i , які відповідають L_i' (тобто, $L_i^1v = \dots = L_i^{j_i}v = L_i'$), $i = 1, 2$. Якщо $j > 1$, то одержуємо НЗУ λ_i для $\{L_i^1, \dots, L_i^{j_i}\}$, і нехай $L_i = L_i^1\lambda_i$, $i = 1, 2$. (Зазначимо, що $L_i^1\lambda_i, \dots, L_i^{j_i}\lambda_i$ будуть тими ж, оскільки λ_i НЗУ). Тоді L_i — літера в склеюванні $C_i\lambda_i$ для C_i . Якщо $j = 1$, то нехай $\lambda_i = \varepsilon$ і $L_i = L_i^1\lambda_i$. Нехай $\lambda = \lambda_1 \cup \lambda_2$. Таким чином, зрозуміло, що L_i' є прикладом L_i . Оскільки L_1' і $\neg L_2'$ уніфікуються, то і L_1 , і $\neg L_2$ теж уніфікуються. Нехай σ — НЗУ для L_1 і $\neg L_2$. Нехай

$$\begin{aligned} C &= ((C_1\lambda)\sigma \setminus L_1\sigma) \cup ((C_2\lambda)\sigma \setminus L_2\sigma) = \\ &= ((C_1\lambda)\sigma \setminus (\{L_1^1, \dots, L_1^{j_1}\}\lambda)\sigma) \cup ((C_2\lambda)\sigma \setminus (\{L_2^1, \dots, L_2^{j_2}\}\lambda)\sigma) = \\ &= (C_1(\lambda\sigma) \setminus \{L_1^1, \dots, L_1^{j_1}\}(\lambda\sigma)) \cup (C_2(\lambda\sigma) \setminus \{L_2^1, \dots, L_2^{j_2}\}(\lambda\sigma)), \end{aligned}$$

C — резольвента C_1 і C_2 . Очевидно, що C' є прикладом C , оскільки

$$\begin{aligned} C' &= (C_1'v \setminus L_2'v) \cup (C_2'v \setminus L_2'v) = \\ &= ((C_1v)v \setminus (\{L_1^1, \dots, L_1^{j_1}\}v)v) \cup ((C_2v)v \setminus (\{L_2^1, \dots, L_2^{j_2}\}v)v) = \\ &= (C_1(vv) \setminus \{L_1^1, \dots, L_1^{j_1}\}(vv)) \cup (C_2(vv) \setminus \{L_2^1, \dots, L_2^{j_2}\}(vv)) \end{aligned}$$

і $\lambda\sigma$ — більш загальний уніфікатор, ніж vv .

Лема доведена.

Д о в е д е н и я т е о р е м и. Нехай множина S виконання (тобто така, що може бути виконана), і $A = \{A_1, A_2, \dots\}$ — множина атомів із S . Нехай T — замкнute семантичне дерево, зображене на рис. 11.1.1. За теоремою Ербрана (варіант 1) T має скінченне замкнute семантичне дерево T' . Якщо T' складається лише з однієї (кореневої) вершини, то 0 повинен належати S , оскільки жодний інший диз'юнкт не може спростовуватись у корені семантичного дерева. Ясно, що в цьому випадку теорема справедлива. Припустимо, що T' складається більше ніж з однієї вершини. Тоді T' має хоча б одну вершину-виведення, бо коли б його не було, то кожна вершина мала хоча б одного нащадка — вершину-спростовування. І тоді можна було б знайти нескінчений ланцюг, який виходить з T' , а це суперечить скінченності T' . Нехай N — вершина-виведення в T' , а N_1 і N_2 — вершини-спростовування, які знаходяться безпосередньо нижче вершини N . Нехай

$$\begin{aligned} I(N) &= \{m_1, m_2, \dots, m_n\}, \\ I(N_1) &= \{m_1, m_2, \dots, m_n, m_{n+1}\}, \\ I(N_2) &= \{m_1, m_2, \dots, m_n, -m_{n+1}\}. \end{aligned}$$

Оскільки N_1 і N_2 — вершини-спростовування, а вершина N — ні, то повинні існувати два такі основні приклади C'_1 і C'_2 диз'юнктів C_1 і C_2 , що C'_1 і C'_2 хибні в $I(N_1)$ і $I(N_2)$ відповідно, але C'_1 і C'_2 не спростовуються в $I(N)$. Значить, C'_1 повинно включати $-m_{n+1}$ і C' повинно включати m_{n+1} . Нехай $L'_1 = -m_{n+1}$ і $L'_2 = m_{n+1}$. Відрізуючи літери L'_1 і L'_2 , ми можемо одержати резольвенту C' для C_1 і C_2 , а саме,

$$C' = (C'_1 \setminus L'_1) \cup (C'_2 \setminus L'_2).$$

C' повинно бути хибним в $I(N)$, оскільки $(C'_1 \setminus L'_1)$ і $(C'_2 \setminus L'_2)$ хибні в $I(N)$. За лемою 11.2.1 існує така резольвента C із C_1 і C_2 , що C' — основний приклад C . Нехай T'' — замкнute семантичне дерево для $(S \cup \{C\})$, одержане з T шляхом вилучення вершини або ребра, що знаходиться нижче першої вершини, і де спростовується резольвента C' . Очевидно, що число вершин в T'' менше, ніж число вершин в T' . Застосовуючи наведений вище процес до T'' , можна одержати іншу резольвенту диз'юнктів із $(S \cup \{C\})$. Підставляючи дану резольвенту в $(S \cup \{C\})$, можна одержати інше замкнute семантичне дерево з меншим числом вершин. Цей процес повторюється, поки не буде породжене замкнute семантичне дерево, що складається з однієї кореневої вершини. А це можливо лише тоді, коли виведений пустий диз'юнкт 0. Отже, існує виведення 0 із S .

Навпаки, нехай тепер існує виведення 0 із S , і нехай $R_1, R_2,$

..., R_k — резольвенти в цьому виведенні. Припустимо, що S виконується на моделі M . Але якщо модель задовільняє диз'юнкти C_1 і C_2 , то вона також повинна задовільняти і будь-яку їх резольвенту (див. теорему 3.3.2). Отже, M задовільняє диз'юнкти R_1, R_2, \dots, R_k . Але це неможливо, оскільки одна із резольвент є 0. Таким чином, S не може виконуватися, що і потрібно було довести.

4. ПРИКЛАДИ ЗАСТОСУВАННЯ МЕТОДУ РЕЗОЛЮЦІЙ

Метод резолюцій легко застосовується для розв'язання різного роду задач, які просто формулюються словами.

1. Деякі пацієнти люблять своїх лікарів. Жоден пацієнт не любить захарів. Значить, жоден лікар не може бути захарем.

Вияснимо, чи має місце висновок, якщо факти вірні. Познайомимо:

$P(x) - x$ — пацієнт;

$D(x) - x$ — лікар;

$Q(x) - x$ — захар;

$L(x, y): x$ любить y .

Тоді факти і висновок можна записати такими формулами:

$F: (\exists x)(P(x) \& (\forall y)(D(y) \rightarrow L(x, y)))$;

$G: (\forall x)(P(x) \rightarrow (\forall y)(Q(y) \rightarrow \neg L(x, y)))$;

$H: (\forall x)(D(x) \rightarrow \neg Q(x))$.

Розглянемо. Зведемо формули F, G, H до стандартної скolemівської форми:

$$\begin{aligned} F: & (\exists x)(P(x) \& (\forall y)(\neg D(y) \vee L(x, y))) = \\ & = (\exists x)(\forall y)(P(x) \& (\neg D(y) \vee L(x, y))). \end{aligned}$$

Вилучивши квантор існування з цієї формули, одержимо таку формулу:

$$F: (\forall y)(P(a) \& (\neg D(y) \vee L(a, y)))$$

Формула G вже знаходиться в скolemівській нормальній формі.

$$\begin{aligned} G: & (\forall x)(\neg P(x) \vee (\forall y)(\neg Q(y) \vee \neg L(x, y))) = \\ & = (\forall x)(\forall y)(\neg P(x) \vee (\neg Q(y) \vee \neg L(x, y))). \end{aligned}$$

Будуємо заперечення $\neg H$ формули H :

$$\neg(\forall x)(\neg D(x) \vee \neg Q(x)) = (\exists x)(\neg(\neg D(x) \vee \neg Q(x))) = (\exists x)(D(x) \& Q(x)).$$

Виконуючи сколемізацію $\neg H$, одержуємо $\neg H = D(a) \& Q(a)$.

Таким чином, маємо такі диз'юнкти:

1) $P(a)$, — посилка;

2) $\neg D(y) \vee L(a, y)$ — посилка;

- 3) $\neg P(x) \vee \neg Q(y) \vee \neg L(x, y)$ — посилка;
 4) $D(a)$, — посилка;
 5) $Q(a)$ — посилка;
 6) $\neg Q(y) \vee \neg L(a, y)$ — ПР із 1), 3);
 7) $\neg D(y) \vee \neg Q(y)$ — ПР із 2), 6);
 8) $\neg Q(a)$ — ПР із 4), 7);
 9) 0 — ПР із 5), 8).

Отже, жоден лікар не є знахарем.

2. Всякий, хто зберігає гроші в банку, одержує відсотки. Якщо не існує відсотків, то ніхто не зберігає гроші в банку.

Нехай:

- $S(x, y)$ — банк x зберігає гроші y ,
 $M(x)$ — x має гроші,
 $I(x)$ — x дає відсотки,
 $E(x, y)$ — x одержує y .

Тоді посилка запишеться у вигляді

$$\forall(x) ((\exists y)(S(x, y) \& M(y)) \rightarrow (\exists y)(I(y) \& E(x, y))),$$

а висновок так:

$$\neg(\exists x) I(x) \rightarrow (\forall x)(\forall y)(S(x, y) \rightarrow \neg M(y)).$$

Р о з в 'я з о к. Перетворюючи дані формули до нормальної форми і заперечуючи висновок, одержуємо такі диз'юнкти:

- 1) $\neg S(x, y) \vee \neg M(y) \vee I(f(x))$ — посилка;
 2) $\neg S(x, y) \vee \neg M(y) \vee E(x, f(x))$ — посилка;
 3) $\neg I(z)$, — заперечення висновку;
 4) $S(a, b)$ — заперечення висновку;
 5) $M(b)$ — заперечення висновку;
 6) $\neg S(x, y) \vee \neg M(y)$, — ПР із 3), 1);
 7) $\neg M(b)$ — ПР із 4), 6);
 8) 0 — ПР із 5), 7).

Отже, висновок доведено.

3. Дано множина формул:

$$A: (\forall x) (C(x) \rightarrow (W(x) \& R(x)));$$

$$B: (\exists x) (C(x) \& Q(x));$$

$$C: (\exists x) (Q(x) \& R(x)).$$

Показати, що формула C — логічний наслідок формул A і B .

Р о з в 'я з о к. Побудуємо для формул A , B і $\neg C$ стандартну скolemівську формулу:

$$\begin{aligned}
 A: & (\forall x) (\neg C(x) \vee (W(x) \& R(x))) = \\
 & = (\forall x) (+C(x) \vee W(x)) \& (\neg C(x) \vee R(x)); \\
 B: & (\exists x) (C(x) \& Q(x)) = C(a) \& Q(a);
 \end{aligned}$$

$$\neg C: \neg(\exists x)(Q(x) \& R(x)) = (\forall x)(\neg(Q(x) \& R(x))) = \\ = (\forall x)(\neg Q(x) \vee \neg R(x)).$$

Таким чином, маємо такі диз'юнкти:

- 1) $\neg C(x) \vee W(x);$
- 2) $\neg C(x) \vee R(x);$
- 3) $C(a);$
- 4) $Q(a);$
- 5) $\neg Q(x) \vee \neg R(x).$

Звідси знаходимо:

- 6) $R(a) \quad \text{— ПР із 3), 2);$
- 7) $\neg R(a) \quad \text{— ПР із 5), 4);$
- 8) $0 \quad \text{— ПР із 7), 6).}$

Отже, формула C є логічним наслідком формул A і B .

5. ПРАВИЛО ПОГЛИНАННЯ

Як ми зазначили, при застосуванні правила резолюції для перевірки того, що дана множина диз'юнктів несуперечна, породжується багато диз'юнктів, які ніякої ролі в доведенні не відіграють. Такі диз'юнкти називаються *зайвими*. Для боротьби із зайвими диз'юнктами існує багато стратегій застосування правила резолюції. Розглянемо одну з них — стратегію *викреслювання*, або *поглинання*.

Означення 11.2.4. Диз'юнкт C називається *піддиз'юнктом* D (або поглинає D) тоді і тільки тоді, коли існує така підстановка σ , що $C\sigma = D$. При цьому D називають *наддиз'юнктом* C .

Приклад 11.2.9

Нехай $C = P(x)$ і $D = P(a) \vee Q(a)$. Якщо $\sigma = \{a/x\}$, то $C\sigma = P(a)$. Оскільки $C\sigma = D$, то C — піддиз'юнкт D . \blacktriangleleft

Стратегія викреслювання полягає в тому, що викреслюються будь-які тавтології і наддиз'юнкти, де це тільки можливо.

Ця стратегія є узагальненням методу Патнема—Девіса, який розглядався в розділі 3, § 3.3. Повнота стратегії викреслювання залежить від того, як викреслювати тавтології і наддиз'юнкти. Стратегія викреслювання буде повною, якщо їх викреслювати таким чином. Спочатку диз'юнкти $\{S^0, S^1, \dots, S^{n-1}\}$ виписуються по порядку; потім обчислюються резольвенти шляхом порівняння кожного диз'юнкта $C_2 \in S^{n-1}$, який записується після C_1 . Коли резольвента знайдена, вона записується в кінець списку, як тільки вона породжується і не є тавтологією або не поглинається

яким-небудь наддиз'юнктом із цього списку. В протилежному випадку вона викреслюється. Доведення повноти цієї стратегії викреслювання пропонується читачеві як проста вправа.

Приклад 11.2.10

Припустимо, нам потрібно довести методом резолюцій, що множина $S = S^0 = \{P \vee Q, \neg P \vee Q, P \vee \neg Q, \neg P \vee \neg Q\}$ суперечна. Зробимо це двома шляхами: а) без застосування стратегії викреслювання; б) із застосуванням цієї стратегії. Маємо:

- a) S^0 : 1) $P \vee Q$,
- 2) $\neg P \vee Q$,
- 3) $P \vee \neg Q$,
- 4) $\neg P \vee \neg Q$,

S^1 :	5) Q	— із 1), 2),
	6) P	— із 1), 3),
	7) $Q \vee \neg Q$	— із 1), 4),
	8) $P \vee \neg P$	— із 1), 4),
	9) $Q \vee \neg Q$	— із 2), 3),
	10) $P \vee \neg P$	— із 2), 3),
	11) $\neg P$	— із 2), 4),
	12) $\neg Q$	— із 3), 4),
S^2 :	13) $P \vee Q$	— із 1), 7),
	14) $P \vee Q$	— із 1), 8),
	15) $P \vee Q$	— із 1), 9),
	16) $P \vee Q$	— із 1), 10),
	17) Q	— із 1), 11),
	18) P	— із 1), 12),
	19) Q	— із 2), 6),
	20) $\neg P \vee Q$	— із 2), 7),
	21) $\neg P \vee Q$	— із 2), 8),
	22) $\neg P \vee Q$	— із 2), 9),
	23) $\neg P \vee Q$	— із 2), 10),
	24) $\neg P$	— із 2), 12),
	25) P	— із 3), 5),
	26) $P \vee \neg Q$	— із 3), 7),
	27) $P \vee \neg Q$	— із 3), 8),
	28) $P \vee \neg Q$	— із 3), 9),
	29) $P \vee \neg Q$	— із 3), 10),
	30) $\neg Q$	— із 3), 11),
	31) $\neg P$	— із 4), 5),
	32) $\neg Q$	— із 4), 6),
	33) $\neg P \vee \neg Q$	— із 4), 7),

- 34) $\neg P \vee \neg Q$ — із 4), 8),
 35) $\neg P \vee \neg Q$ — із 4), 9),
 36) $\neg P \vee \neg Q$ — із 4), 10),
 37) Q — із 5), 7),
 38) Q — із 5), 9),
 39) 0 — із 5), 12);

- 6) S^0 : 1) $P \vee Q$,
 2) $\neg P \vee Q$,
 3) $P \vee \neg Q$,
 4) $\neg P \vee \neg Q$,

- S^1 : 5) Q — із 1), 2),
 6) P — із 1), 3),
 7) $\neg P$ — із 2), 4),
 8) $\neg Q$ — із 3), 4),
 9) 0 — із 5), 8).

Як бачимо, другий шлях доведення суперечності даної множини диз'юнктів набагато економніший. ▲

Для того щоб використовувати стратегію викреслювання, необхідно вміти відповідати на питання, чи є диз'юнкт тавтологією або піддиз'юнктом деякого іншого диз'юнкта. Перевірка того, що диз'юнкт — тавтологія, зводиться, очевидно, до перевірки існування в ньому контрапарних пар. Перевірка ж того, що даний диз'юнкт є піддиз'юнктом, не зовсім проста.

Нехай C і D — диз'юнкти і $\sigma = \{x_1/a_1, \dots, x_n/a_n\}$, де x_1, \dots, x_n — змінні, які зустрічаються в D , і a_1, \dots, a_n — нові різні константи, які не зустрічаються ні в C , ні в D . Покладемо $D = L_1 \vee L_2 \vee \dots \vee L_m$, тоді $D\sigma = L_1\sigma \vee L_2\sigma \vee \dots \vee L_m\sigma$. Зазначимо, що $D\sigma$ — основний диз'юнкт, $\neg D\sigma = \neg L_1\sigma \& \neg L_2\sigma \& \dots \& \neg L_m\sigma$. Наведений нижче алгоритм перевіряє, чи є диз'юнкт C піддиз'юнктом диз'юнкта D . При цьому диз'юнкт D подається у вигляді

$$\neg D\sigma = \neg L_1\sigma \& \neg L_2\sigma \& \dots \& \neg L_m\sigma.$$

Алгоритм поглинання (C, D)

1. $W = \{\neg L_1\sigma, \dots, \neg L_m\sigma\}$.
2. Покласти $k = 0$ і $U^0 = \{C\}$.

3. Якщо U^k включає 0, то кінець; C — піддиз'юнкт D . У протилежному випадку покладаємо $U^{k+1} = \{\text{резольвента } C_1 \text{ і } C_2 \mid C_1 \in U^k \text{ і } C_2 \in W\}$.

4. Якщо U^{k+1} — пуста множина, то кінець; C не є піддиз'юнктом D . У протилежному випадку покладаємо $k = k + 1$ і переходимо до кроку 3.

Зазначимо, що в цьому алгоритмі кожний диз'юнкт в U^{k+1}

має на одну літеру менше, ніж диз'юнкт в U^k , з якого він був одержаний. Значить, в послідовності U^0, U^1, \dots повинна врешті-решт з'явитися або пуста множина, або множина, що має 0.

Теорема 11.2.3. Алгоритм поглибання коректний, тобто C є піддиз'юнктом D тоді і тільки тоді, коли цей алгоритм закінчує роботу на кроці 3.

Доведення. Якщо C — піддиз'юнкт D , то існує така підстановка σ , що $C\sigma \subseteq D$. Значить, $C(\sigma * v) \subseteq Dv$. Таким чином, літери $C(\sigma * v)$ можуть бути відрізані шляхом використання одиничних основних диз'юнктів в W . Але $C(\sigma * v)$ є прикладом C . Отже, літери в C можуть бути відрізані теж шляхом використання одиничних диз'юнктів в W . А це значить, що ми зрештою знайдемо U , яке включає 0. Таким чином, алгоритм закінчить свою роботу на кроці 3.

Навпаки, якщо алгоритм завершується на кроці 3, то отримуємо спростування $B_0, R_1, B_1, R_2, \dots, B_{r-1}, R_r$, де B_0, \dots, B_r — одиничні диз'юнкти із W , R_i — резольвента C і B_0 , а R_i — резольвента R_{i-1} і B_{i-1} для $i = 2, \dots, r$.

Нехай σ_0 — НЗУ, одержаний внаслідок застосування правила резолюції до C і B_0 , а σ_i — НЗУ для R_i і B_i , $i = 1, \dots, r$. Тоді $C(\sigma_0 * \sigma_1 * \dots * \sigma_r) = \{\neg B_0, \neg B_1, \dots, \neg B_r\} \subseteq Dv$. Нехай $\lambda = \sigma_0 * \sigma_1 * \dots * \sigma_r$. Тоді $C\lambda \subseteq Dv$. Нехай σ — підстановка, одержана з λ заміною в кожному компоненті λ a_i на x_i для $i = 1, \dots, n$. Тоді $C\delta \subseteq D$. Отже, C — піддиз'юнкт D , що і потрібно було довести.

Приклад 11.2.11

Нехай $C = \neg P(x) \vee Q(f(x), a)$ і $D = \neg P(h(y)) \vee Q(f(h(y)), a) \vee \neg P(z)$. Чи є C піддиз'юнктом D ?

Розв'язок. y і z — змінні в D . Нехай $\sigma = \{y/b, z/c\}$. Зauważимо, що b і c відсутні в C і D . Тоді $D\sigma = \neg P(h(b)) \vee Q(f(h(b)), a) \vee \neg P(c)$. Таким чином, $\neg D\sigma = P(h(b)) \wedge \neg Q(f(h(b)), a) \wedge P(c)$.

Значить,

$$\begin{aligned} W &= \{P(h(b)), \neg Q(f(h(b)), a), P(c)\}, \\ U^0 &= \{\neg P(x) \vee Q(f(x), a)\}. \end{aligned}$$

Оскільки U^0 не включає 0, то внаслідок застосування правила резолюції, одержуємо

$$U^1 = \{Q(f(h(b)), a)\} \text{ (НЗУ } \sigma = \{x/h(b)\}\text{)}.$$

Оскільки U^1 не пуста і не містить 0, то знову в результаті застосування правила резолюції одержуємо $U^2 = \{0\}$.

Оскільки U^2 містить 0, то алгоритм закінчує свою роботу, і ми робимо висновок, що C — піддиз'юнкт D . \blacktriangleleft

Контрольні питання

1. Що називається ербранівським універсумом?
2. Що називається ербранівським базисом?
3. Що являє собою H -інтерпретація?
4. Що називається семантичним деревом?
5. Що таке повне і закрите семантичне дерево?
6. Яка підстановка називається уніфікатором?
7. Які Ви знаєте алгоритми уніфікації?
8. Який уніфікатор називається НЗУ?
9. Сформулюйте теорему про повноту методу резолюцій.
10. Який вигляд має формула в сколемівській нормальній формі?

Задачі і вправи

1. Довести, що:

а) основний приклад C' диз'юнкта C виконується при інтерпретації h тоді і тільки тоді, коли існує основна літера L' в C' , така, що L' виконується і в h ;

б) диз'юнкт C виконується при інтерпретації h тоді і тільки тоді, коли кожний основний приклад C виконується при інтерпретації h ;

в) диз'юнкт C спростовується інтерпретацією h тоді і тільки тоді, коли існує хоча б один основний приклад C' диз'юнкта C , такий, що C' не виконується при інтерпретації h ;

г) множина диз'юнктів S суперечна тоді і тільки тоді, коли для кожної інтерпретації h існує хоча б один основний приклад C' деякого диз'юнкта C із S , такий, що C' не виконується при інтерпретації h .

2. Нехай $S = \{P(f(x)), a, g(f(x)), b\}$. Знайти:

а) H_0 і H_1 ;

б) всі основні приклади S на H_0 ;

в) всі основні приклади S на H_1 .

3. Нехай $S = \{P(x), \neg P(f(y))\}$. Побудувати H_0, H_1, H_2, H_3 для S . Чи можна знайти інтерпретацію, яка задовільняє S ? Якщо можна, то побудуйте одну. Якщо не можна, то чому?

4. Нехай $S = \{P, \neg P \vee Q, \neg Q\}$. Побудувати:

а) множину атомів S ;

б) повне семантичне дерево S ;

в) закрите семантичне дерево S .

5. Покажіть, що наведені нижче твердження є логічними наслідками таких фактів:

а) Посилка: студенти є громадянами країни.

Висновок: голоси студентів на виборах є голосами громадян;

б) Посилка: всякий, хто знаходиться в здоровому глузді, може розуміти математику. Жоден із синів Гегеля не може розуміти математику. Ті, що з'їхали з глузду, не можуть брати участі у виборах.

Висновок: ніхто з синів Гегеля не може брати участі у виборах;

г) Посилка: всякий предок предка даної людини є предком тієї ж людини, і жодна людина не може бути сама собі предком.

Висновок: існує деякий індивід, який не має предків.

6. За допомогою методу резолюцій покажіть, що такі множини диз'юнктів не виконуються:

$$P \vee Q \vee R, \neg P \vee R, \neg Q, \neg R.$$

7. Нехай $\sigma = \{x/a, y/b, z/g(x, y)\}$ — підстановка і $E = P(f(x), z)$. Знайдіть $E\sigma$.

8. Нехай $\sigma = \{x/a, y/f(z), z/y\}$ і $\sigma_1 = \{x/b, y/z, z/g(x)\}$. Знайдіть добуток $\sigma * \sigma_1$.

9. Доведіть, що для будь-яких підстановок σ, τ, δ має місце рівність $(\sigma * \tau) * \delta = \sigma * (\tau * \delta)$.

10. Визначіть, які з наведених нижче множин мають НЗУ, і якщо такий НЗУ є, то знайдіть його:

а) $W = \{Q(a), Q(b)\};$

б) $W = \{Q(a, x), Q(a, a)\};$

в) $W = \{Q(a, x, f(x)), Q(a, y, y)\};$

г) $W = \{Q(x, y, z), Q(u, h(u, v), u)\};$

д) $W = \{P(x_1, g(x_1), x_2, h(x_1, x_2), x_3, k(x_1, x_2, x_3)),$

$P(y_1, y_2, e(y_2), y_3, f(y_2, y_3), y_4)\};$

11. Знайдіть всі можливі резольвенти (зрозуміло, якщо вони існують) для таких пар диз'юнктів:

а) $C = \neg P(x) \vee Q(x, b), D = P(a) \vee Q(a, b);$

б) $C = \neg P(x) \vee Q(x, x), D = \neg Q(a, f(a));$

в) $C = \neg P(x, y, u) \vee \neg P(y, z, v) \vee \neg P(x, v, w) \vee P(u, z, w),$

$D = P(g(x, y), x, y).$

12. Доведіть за допомогою правила резолюцій, що формула $\neg B \rightarrow \rightarrow \neg A$ є логічним наслідком формули $A \rightarrow B$.

ОСНОВНІ ПОНЯТТЯ ТЕОРИЇ ПРОГРАМНИХ ІНВАРІАНТІВ

Алгебри відіграють важливу роль у задачах аналізу, верифікації і оптимізації програм, розв'язок яких дає можливість підвищити ефективність програмного забезпечення. Перелічені задачі в загальному випадку не є алгоритмічно розв'язними, але в окремих випадках їх розв'язок значно полегшується при умові, що нам відомі інваріантні співвідношення в кожному стані програми. Задача пошуку такого роду співвідношень — це окрема проблема, а теорія, в якій ця проблема вивчається, називається *теорією програмних інваріантів*. У цьому розділі розглядаються методи пошуку програмних інваріантів для програм з простими змінними. Основною моделлю програм виступають *U-Y*-схеми програм над пам'яттю.

§ 12.1. ІНВАРІАНТИ В СТАНАХ *U-Y*-ПРОГРАМ І МОВИ ІНВАРІАНТІВ

1. ОЗНАЧЕННЯ ПРОГРАМНОГО ІНВАРІАНТА

Нехай A — *U-Y*-схема програми над пам'яттю $R = \{r_1, \dots, r_m\}$, інтерпретована на області даних D (*U-Y*-програма), і L — мова, якою записуються твердження про властивості інформаційного середовища B . Відносно мови L будемо вважати, що всяке її слово (яке називатиметься також умовою) можна виразити формулою $\Phi(r)$ числення предикатів першого порядку, яка містить вільні змінні з кортежу $r = (r_1, \dots, r_m)$ і інтерпретована на області даних D . Сигнатуря цього числення містить всі символи сигнатур Ω і Π .

Означення 12.1.1. Умова $\Phi(r)$ називається *інваріантом стану* а *U-Y*-програми A *відносно* умови $u(r)$, якщо вона істинна при

кожному проходженні стану a в процесі виконання програми A для тих і тільки тих початкових станів пам'яті з B , на яких $u(r)$ істинна. Умова $u(r)$ називається *початковою*. Якщо початкова умова є тотожністю на D , то $\Phi(r)$ будемо називати просто *інваріантом стану a*.

Зміст інваріанта $\Phi(r)$ стану a відносно умови $u(r)$ характеризує множина $B(a) \subseteq B$ станів інформаційного середовища, які досягаються в стані a заданої U - Y -програми на допустимих шляхах, що починаються на станах пам'яті, які задовольняють початкову умову $u(r)$. Схема побудови множини $B(a)$ така. Будується множина S всіх допустимих шляхів із a_0 в a , які реалізуються для початкових станів із $B_u \subseteq B$ (характеристична множина умови $u(r)$). За кожним шляхом $s = (a_0, b_0), \dots, (a_k, b_k), (a_{k+1}, b_{k+1})$, де $k > 0$, $a_{k+1} = a$, відновлюється операторна історія — послідовність y_1, \dots, y_k операторів, якими відмічені переходи на цьому шляху. Розглядаючи після цього композицію $y_s = y_1 * \dots * * y_k$ (а це теж оператор присвоювання над R), будуємо множину $B_s = y_s(B_u)$, після чого шукана множина $B(a)$ розглядається як B_u . Але ця схема неконструктивна як внаслідок неконструктивності множини S , так і внаслідок того, що множини виду B_s можуть не бути характеристичними для умов мови L . Виходом з такого положення є апроксимація множини S деякою надмножиною шляхів S' , яка хоча і включає “зайві” (недопустимі) шляхи, але структура її простіша, так і апроксимація множин виду B_s їх підмножинами B'_s , характеристичними для деяких умов із L . Обидві апроксимації ведуть до зменшення множини $B(a)$, оскільки зменшується кожний член перетину $\bigcap_{s \in S} B_s$, і добавляються нові

його члени. З урахуванням звуження множини $B(a)$ з неконструктивної $\bigcap_{s \in S} B_s$ до деякого наближення знизу можна говорити

про максимальний інваріант стану a як про умову $\Phi(r)$ мови L , таку, що $B_u \subseteq B(a)$, і для всякого іншого інваріанта $\Phi'(r)$ має місце $\Phi(r) \Rightarrow \Phi'(r)$, де \Rightarrow означає відношення “бути наслідком”.

Побудова інваріантів виконується *генератором інваріантів* у мові L . Поняття генератора включає три компоненти:

- функцію $ef : L \times U \times Y \rightarrow L$ — “ефект оператора”;
- структуру напівструктур на множині умов із L ;
- ітераційний алгоритм.

Функція ef за умовами u' і u із L , істинними перед виконанням оператора $u \in Y$, будує умову $ef(u', u, y)$, яка істинна на перетвореному оператором u стані пам'яті. Інколи розглядаємо простіший варіант функції ef , коли серед її аргументів немає

умови u , яка обумовлює перехід u/y . Слід зазначити, що з означення функції ef випливає насамперед її монотонність відносно першого аргументу, тобто з того, що множина умов N — логічний наслідок множини умов N' , випливає, що $\text{ef}(N, u, y) = \text{ef}(N', u, y)$ — логічний наслідок $\text{ef}(N', u, y)$. Дійсно, $\text{ef}(N, u, y)$ виконанна (тобто така, що може бути виконаною) на послідовності $y(b)$ тоді і тільки тоді, коли N і u виконанні на послідовності b із B . Але тоді $\text{ef}(N', u, y)$ виконанна на всякій послідовності $y(b)$, на якій виконанна $\text{ef}(N, u, y)$, оскільки N — наслідок N' .

Операція \wedge взяття нижньої грані на множині умов мови L , взагалі кажучи, відрізняється від кон'юнкції умов: наприклад, якщо L — мова лінійних рівнянь і $u_1 = ((x = 2) \& (y = 3))$, $u_2 = ((x = 3) \& (y = 2))$, то нижня грань дорівнює $(x + y = 5) \& (|x - y| = 1)$, хоча $u_1 \& u_2 = 0$. І останнє, ітеративний алгоритм, відправним пунктом якого є деяка початкова розмітка (коли відміткою стану a є умова $u(r)$, а відмітки решти вершин — тотожно істинні умови), буде послідовність відміток, які уточнюються до моменту одержання деякої стабільної розмітки, що відповідає системі інваріантів.

Розглянемо, як виконується ітераційний крок. Нехай задана деяка розмітка станів U - Y -програми, $N(a) \in L$ — поточна відмітка стану a і $z: V \rightarrow \{0, 1\}$ — функція, яка дає можливість відділити стани, для яких відмітки обновились, від решти станів. Алгоритм завершує роботу, якщо $(\forall a \in V)(z(a) = 0)$. Якщо ж ця умова не виконується, то вибирається стан a , такий, що $z(a) = 1$. Після цього для кожного переходу з даного стану (а їх може бути один або два) будуємо уточнення. Уточнення для переходу $a \rightarrow a'$, відміченого парою u/y , полягає в тому, що обчислюється ефект $\text{ef}(N(a), y)$, а потім — нижня грань $\text{ef}(N(a), y) \wedge N(a')$, і, нарешті, якщо ця нижня грань відмінна від $N(a')$, то $z(a')$ кладемо рівним 1. Уточнення переходів, які виходять із стану a , супроводжується зміною $z(a)$ з 1 на 0. Задавши так ітераційний крок, загальний для всіх алгоритмів даного класу, зауважимо, що в деяких ситуаціях в станах-розгалуженнях уточнення виконується не по обох переходах, а лише по одному. Для цього необхідно, щоб умови u або $\neg u$, які визначають вибір альтернативи в розгалуженні, були наслідком поточної відмітки $N(a)$. Скінченість ітераційного процесу, як правило, гарантують умови обриву спадних ланцюгів у напівструктурах мови L , що, на жаль, може впливати на якість функції ef .

Якщо описаний алгоритм завершився, то функція ef дає сукупність інваріантів в станах U - Y -програми. Цю сукупність також будемо позначати $\{N_a\}$, де $a \in A$. Виникає питання про якість

одержаної системи інваріантів. Для цього за історією роботи ітераційного алгоритму будується множина шляхів, яка ним підтримується і яка апроксимує множину допустимих шляхів. Для кожного стану a' U - Y -програми до моменту стабілізації розмітки була пройдена лише скінчenna множина шляхів із початкового стану a в стан a' , і кожному з них відповідає своя операторна історія. Скінченну сукупність цих шляхів позначимо $Q(a')$. Далі, множину операторних історій, які відповідають шляхам із a' в a , опишемо регулярним виразом, який позначимо $T(a', a)$. Зауважимо, що для кожного стану a , з якого ведуть переходи за умовами u і $\neg u$, перевіряється, чи u і $\neg u$ логічними наслідками із $N(a)$, і якщо одна з цих умов є наслідком із $N(a)$, то переход, що відповідає другій умові, ігнорується при синтезі $T(a', a)$. Після того, як побудовані всі множини $Q(a')$ і $T(a', a)$, апроксимація множини S допустимих шляхів із a_0 в a будується як

$$\bigcup_{a \in A} \text{Path}(a) = Q(a') * T(a', a).$$

2. МЕТОДИ ПОШУКУ ПРОГРАМНИХ ІНВАРІАНТІВ

Перейдемо тепер до більш традиційної термінології, яка прийнята в теорії програмних інваріантів — теоретико-множинної. Оскільки N і $\text{ef}(N, u, y)$ — деякі предикати на множині D , то їх можна розглядати як відношення на D , які визначаються цими предикатами. Тоді булеван $\mathbf{B}(L)$ зручно представляти у вигляді структури відносно операцій перетину \cap і об'єднання \cup , яка включає нуль \emptyset і одиницю L . Вирази $\text{ef}(N, u, y) \cap (\cup) \text{ef}(N', u', y')$ у цьому випадку розглядають як перетин (об'єднання) відповідних відношень на D , а те, що множина $\text{ef}(N, u, y)$ є логічним наслідком множини формул $\text{ef}(N', u, y)$ — як теоретико-множинне включення $\text{ef}(N, u, y) \subseteq \text{ef}(N', u, y)$. Надалі користувати-мося цими позначеннями.

Число різних можливих шляхів у програмі (при умові існування в ній хоча б одного циклу) може бути нескінченим, і тоді процес побудови умови стану a теж може стати нескінченим. Незважаючи на це, нехай a_1, \dots, a_k — всі стани U - Y -програми A , пов'язані переходами (a_i, u_i, y_i, a) зі станом a , і N_i — множина інваріантів стану a відносно деякої умови. Тоді очевидно, що $\text{ef}(N_i, u_i, y_i)$ буде інваріантом стану a відносно тієї ж початкової умови. Цей простий факт служить відправним пунктом для побудови двох ітераційних методів генерації інваріантів [22].

У першому з них, названому *методом нижньої апроксимації* (МНА), ітераційний процес задається рекурентним співвідношенням

$$N_a^{(n)} = \bigcap_{(a', u, y, a) \in S} \text{ef}(N_a^{(n-1)}, u, y), \quad n > 0, \quad a, a' \in A, \quad (12.1.1)$$

а початкове наближення $\{N_a^{(0)}\}$ — рівняннями $N_a^{(0)} = \{u\}$ і $N_a^{(0)} = \emptyset$ для $a \neq a_0$. Із властивості монотонності функції ef випливає, що

$$N_a^{(0)} \subseteq N_a^{(1)} \subseteq \dots \subseteq N_a^{(m)} \subseteq \dots$$

незалежно від стану a . Вказаний ітераційний процес може завершитися через скінченне число кроків внаслідок стабілізації послідовностей N_a для всіх $a \in A$ або продовжуватися нескінченно, але перевага цього методу полягає в тому, що, не чекаючи стабілізації процесу обчислень, їх можна перервати, оскільки всяка множина N_a включається в множину інваріантів стану a .

У другому методі — *методі верхньої апроксимації* (МВА), ітераційний процес задається рекурентним співвідношенням

$$N_a^{(n)} = N_a^{(n-1)} \cap \left(\bigcap_{(a', u, y, a) \in S} \text{ef}(N_{a'}, u, y) \right), \quad n > 0, \quad a, a' \in A, \quad (12.1.2)$$

а початкове наближення визначається рівнянням $N_{a_0}^{(0)} = \{u\}$ і деякою сукупністю простих шляхів, що покривають всю множину станів U - Y -програми A . Обчислення початкового наближення виконується для всіх таких шляхів, починаючи з $N_a^{(0)}$: якщо для деякого $a \in A$ уже відомо $N_a^{(0)}$, перехід (a, u, y, a') належить одному з шляхів заданої системи, а $N_{a'}^{(0)}$ ще невідоме, то покладаємо $N_{a'}^{(0)} = \text{ef}(N_a, u, y)$. Із співвідношення (12.1.2) випливає, що для всякого $a \in A$ виконуються включення $N_a^{(0)} \supseteq N_a^{(1)} \supseteq \dots$ і, значить, шукана сукупність інваріантів може бути одержана тільки після стабілізації ітераційного процесу. Оскільки процес пошуку інваріантів може бути нескінченим, то це є недоліком методу МВА, який у випадку результативного завершення генерує повніші системи інваріантів, ніж метод МНА.

Дослідження методів генерації інваріантів показує, що скінченність процесу пошуку інваріантів тісно пов'язана з мовою умов L . Найпоширеніші мови умов — мови типу рівностей і нерівностей, оскільки практично всяка мова програмування включає в себе предикати рівності і нерівності. Розробка методів пошуку інваріантів для цих мов пов'язана зі значними труднощами.

Дійсно, формули (12.1.1) і (12.1.2) вимагають алгоритмічного і, по можливості, ефективного розв'язку задач перетину множин співвідношень, перевірки, чи є співвідношення наслідком заданої множини співвідношень тощо. Розв'язок перелічених задач уже для цих мов вимагає застосування досить складних методів сучасної комутативної алгебри, в чому можна переконатися з наступного розділу. Таким чином, використання занадто багатої мови умов не дає можливості одержувати досить повний і вичерпний опис інваріантів. У зв'язку з цим доводиться накладати певні обмеження на мову умов L . Розглянемо деякі з них.

§ 12.2. МОВА РІВНОСТЕЙ. ОСНОВНІ ЗАДАЧІ

1. ОСНОВНІ ЗАДАЧІ ТЕОРІЇ ПРОГРАМНИХ ІНВАРІАНТІВ

Нехай A — U - Y -програма з множиною змінних $R = \{r_1, \dots, r_m\}$, яка розглядається над алгеброю даних (D, Ω) , $K(\Omega, Eq)$ — клас алгебр, до якого належить алгебра (D, Ω) і який визначається множиною тотожних співвідношень Eq , а $T(\Omega, R)$ — вільна алгебра термів над R з класу $K(\Omega, Eq)$.

Розглянемо проблему пошуку інваріантів для мови L , яка складається з умов типу рівностей $g(r) = h(r)$, де $g(r), h(r) \in T(\Omega, R)$, $r = (r_1, \dots, r_m)$ (тобто L не враховує умов із множини U).

Нехай M — деяка множина рівностей. Кожну рівність можна розглядати як пару термів, а множину M — як бінарне відношення на множині термів $T(\Omega, R)$.

Алгебраїчним замиканням множини M відносно Eq називається найменша множина $C(M)$, яка включає в себе рефлексивне, симетричне і транзитивне замикання M , всі тотожності з Eq , і для всякої n -арної операції ω разом з парами $(g_1(r), q_1(r)), \dots, (g_n(r), q_n(r))$ включає пару $(\omega(g_1(r), \dots, g_n(r)), \omega(q_1(r), \dots, q_n(r)))$. Множина M називається *алгебраїчно замкнutoю*, якщо $C(M) = M$.

Теорема 12.2.1. $C(M) = M$ тоді і тільки тоді, коли M — конгруентність на $T(\Omega, R)$.

Доведення очевидним чином випливає з означення конгруентності і алгебраїчного замикання.

Підмножина P алгебраїчно замкнutoї множини M називається *алгебраїчним базисом* M , якщо $C(P) = M$.

Нехай M — алгебраїчно замкнuta множина рівностей. Відповідну фактор-алгебру будемо позначати $T(\Omega, R)/M$, її елементи — $t \pmod{M}$, $t \in T(\Omega, R)$, а рівність термів — у вигляді $t = t' \pmod{M}$. З кожним оператором присвоювання $y = (r_1 := t_1(r), \dots, r_m :=$

$:= t_m(r))$ ($t_i \in T(\Omega, R)$) і алгебраїчно замкнутою множиною M пов'яжемо гомоморфізм

$$h_y : T(\Omega, R) \rightarrow T(\Omega, R)/M,$$

покладаючи $h_y(r_i) = t_i \pmod{M}$.

Конгруентність M на $T(\Omega, R)$ називається **нормальнюю**, якщо вона — ядро ендоморфізму алгебри $T(\Omega, R)$, тобто $T(\Omega, R)/M$ ізоморфна підалгебрі алгебри $T(\Omega, R)$.

Нехай $\text{ef}(M, y)$ (звуження $\text{ef}(M, u, y)$) означає множину рівностей $t(r) = t'(r)$, таких, що $t(t_1, \dots, t_m) = t'(t_1, \dots, t_m) \in M$. Очевидно, що $\text{ef}(M, y)$ — конгруенція і $\text{ef}(M, y) = \ker(h_y)$, де

$$h_y : T(\Omega, R) \rightarrow T(\Omega, R)/M$$

— гомоморфізм, заданий деяким оператором y .

Лема 12.2.1. $\text{ef}(\text{ef}(M, y), y') = \text{ef}(M, yy')$.

Доведення. Дійсно, нехай $y = (r_1 := t_1(r), \dots, r_m := t_m(r))$, $y' = (r_1 := t'_1(r), \dots, r_m := t'_m(r))$. Тоді $yy' = y'' = (r_1 := t'_1(t_1(r), \dots, t_m(r)), \dots, r_m := t'_m(t_1(r), \dots, t_m(r)))$, і якщо $t(r) = t'(r) \in \text{ef}(\text{ef}(M, y), y')$, то $t(t'_1, \dots, t'_m) = t'(t'_1, \dots, t'_m) \in \text{ef}(M, y)$ і $t(t'(t_1, \dots, t_m), \dots, t'_m(t_1, \dots, t_m)) = t'(t'_1(t_1, \dots, t_m), \dots, t'_m(t_1, \dots, t_m)) \in M$. Останнє означає, що $t(r) = t'(r) \in \text{ef}(M, yy') = \text{ef}(M, y'')$. Обернене включення доводиться аналогічно.

Теорема 12.2.2. Якщо M — нормальна конгруентність, то $\text{ef}(M, y)$ — теж нормальна конгруентність.

Доведення. Дійсно, якщо M — нормальна конгруентність, тобто $M = \ker(h)$, то $M = \text{ef}(0, y')$, де $y' = (r_1 := h(r_1), \dots, r_m := (r_m))$ і $\text{ef}(M, y) = \text{ef}(\text{ef}(\emptyset, y'), y) = \text{ef}(\emptyset, y'')$ — нормальна конгруентність за лемою 12.2.2.

Функція ef називається **дистрибутивною**, якщо для будь-яких M і M' виконується рівність $\text{ef}(M \cap M', y) = \text{ef}(M, y) \cap \text{ef}(M', y)$.

Теорема 12.2.3. Якщо множини M і M' алгебраїчно замкнуті, то функція ef дистрибутивна.

Доведення. $t = t' \in \text{ef}(M \cap M', y) \Leftrightarrow y(t) = y(t') \pmod{(M \cap M')} \Leftrightarrow y(t) = y(t') \pmod{M}$ і $y(t) = y(t') \pmod{M'} \Leftrightarrow t = t' \in \text{ef}(M, y) \cap \text{ef}(M', y)$.

Зauważимо, що умова дистрибутивності функції ef сильніша за умову монотонності, оскільки з дистрибутивності функції ef випливає її монотонність.

Дійсно, якщо $M \subseteq M'$, то внаслідок дистрибутивності функції ef можна записати $\text{ef}(M \cap M', y) = \text{ef}(M', y) = \text{ef}(M, y) \cap \text{ef}(M', y)$. Звідси випливає, що $\text{ef}(M, y) \subseteq \text{ef}(M', y)$.

Припустимо, що нам відомо, як будувати множину $\text{ef}(M, y)$ або її алгебраїчний базис і знаходити перетин таких множин або

алгебраїчний базис цього перетину. Тоді, користуючись однією з формул (12.1.1) чи (12.1.2), можна організувати процес пошуку інваріантів у станах програми, виходячи з деяких початкових алгебраїчно замкнутих множин, які поставлені у відповідність станам програми, і, повторюючи його доти, доки множини $\text{ef}(M, y)$ в даних станах не перестануть змінюватися (стабілізуються). Зauważимо, що коли множини $\text{ef}(M, y)$ мають скінченні базиси, то із стабілізації множин $\text{ef}(M, y)$ випливає стабілізація їх базисів і навпаки. Отже, проблема побудови множин інваріантів у станах U - Y -програм над заданою алгеброю даних $T(D, R)$ зводиться до таких основних підзадач.

Задача про співвідношення. Побудувати за заданою множиною рівностей M (або її алгебраїчним базисом) і оператором $y \in Y$ множину $\text{ef}(M, y)$ (або її алгебраїчний базис).

Задача про перетин. За заданими множинами $\text{ef}(M, y)$ і $\text{ef}(M', y)$ побудувати множину $\text{ef}(M, y) \cap \text{ef}(M', y)$ (або її алгебраїчний базис).

Задача про стабілізацію. Показати, що процес побудови множин $\text{ef}(M, y)$ (або їх алгебраїчних базисів), які відповідають станам програми, стабілізується.

2. АЛГОРИТМИ ПОШУКУ ПРОГРАМНИХ ІНВАРІАНТІВ

При визначенні методу МВА процес обчислення інваріантів залежить, взагалі кажучи, від вибору сукупності простих шляхів, які задають початкове наближення. Аналогічна ситуація має місце і при переході від паралельного способу обчислення наближень за формулами (12.1.1) і (12.1.2), коли на кожному кроці ітерації обчислення виконуються одночасно для всіх станів, до послідовного, коли в даний момент для різних станів розглядаються, взагалі кажучи, різні наближення. Послідовні обчислення наближень дозволяють суттєво зменшити об'єм обчислень, оскільки для частини станів на деяких ітераціях взагалі не буде ніяких змін. Наприклад, якщо існує шлях із стану a в стан a' і не існує шляху із a' в a , то множина інваріантів у стані a ніяк не залежить від відповідної множини для стану a' , і процеси обчислення інваріантів для таких станів зручно рознести, обчислюючи їх спочатку для a і тільки потім — для a' .

Нижче наводяться послідовні алгоритми нижньої і верхньої апроксимацій. Відносно початкової U - Y -програми будемо допускати, по-перше, що всі її стани, можливо, за винятком початкового a_0 і заключного a^* , є розгалуженнями або злиттями — цього можна досягти в результаті множення операторів при-

своювання на лінійних ланках, і, по-друге, з кожним станом a асоціюється множина співвідношень N_a . Нехай S — множина переходів U - Y -програми і $\text{Ps}(a) = \{a' \in A \mid (a, u, y, a') \in S\}$. В наведених нижче алгоритмах N_a , N , C — змінні типу множина, значення яких зрозумілі з алгоритмів, N_0 — початкова множина співвідношень, $v(1 : |A|)$ — масив логічних значень.

У цих позначеннях алгоритми пошуку інваріантів можна записати так:

МНА(A , N_0)

початок

/* перший етап роботи алгоритму МНА */

$N_{a_0} := N_0$;

$C := A \setminus \{a_0\}$;

для всіх a із C **виконати**

$N_a := \emptyset$;

кц;

/* другий етап роботи алгоритму МНА */

поки $C \neq \emptyset$ **виконати**

взяти a із C ;

$N := N_a$;

$I := 1$;

для всіх $(a', u, y, a) \in S$ **виконати**

якщо $N = \emptyset$ **то**

якщо $I = 1$ **то**

$N := \text{ef}(N_a, y)$;

$I := 2$;

інакше

вийти з циклу;

кя;

інакше

$N := N \cap \text{ef}(N_a, y)$;

кя;

кц;

якщо $N \neq N_a$ **то**

$N_a := N$;

$C := C \cup \text{Ps}(a)$;

кя;

кц;

кінець.

початок

/* перший етап роботи алгоритму MBA */

$N_{a_0} := N_0;$

для всіх a із $A \setminus \{a_0\}$ виконати

$v(a) := 0;$

$N_a := 1;$

кц;

$C := \{a_0\};$

$v(a_0) := 1;$

поки $C \neq \emptyset$ виконати

взяти a' із C ;

для всіх $(a', u, y, a) \in S$ виконати

якщо $v(a) = 0$ то

$N_a := \text{ef}(N_{a'}, y);$

$C := C \cup \{a\};$

$v(a') := 1;$

кя;

кц;

кц;

/* другий етап роботи алгоритму MBA */

$C := A \setminus \{a_0\};$

поки $C \neq \emptyset$ виконати

взяти a із C ;

$N := N_a;$

для всіх $(a', u, y, a) \in S$ виконати

якщо $N = \emptyset$ то

вийти з циклу;

інакше

$N := N_a \cap \text{ef}(N_{a'}, y);$

кя;

кц;

якщо $N \neq N_a$ то

$N_a := N;$

$C := C \cup \{\text{Ps}(a)\};$

кя;

кц;

кінець.

Семантика оператора **взяти a із C** така: фіксується елемент a із множини C , який при цьому вилучається із вказаної множини, а оператор **вийти з циклу** означає закінчення циклу, в якому він знаходиться. Слова в дужках типу /* і */ — це коментарі. Стра-

тегія вибору елементів з множини C , взагалі кажучи, довільна, але вважається, що завжди останнім елементом, який вибирається із C , є заключний стан a^* . Множина простих шляхів, за якою визначається початкове наближення, серед вхідних параметрів не фіксується.

Разом з наведеними алгоритмами можна розглядати їх версії, орієнтовані на роботу з алгебраїчними базисами. Вони відрізняються тим, що значенням вхідного параметра N є не сама множина співвідношень, а її базис, і тоді розглядається не операція перетину множин співвідношень, а операція побудови базису їх перетину.

Встановимо тепер деякі загальні властивості наведених вище алгоритмів.

Теорема 12.2.4. Якщо в алгебрі $T(\Omega, R)$ виконується умова обриву спадних ланцюгів нормальних конгруентностей, множина N і перетин нормальних конгруентностей є нормальними конгруентностями, то алгоритм МВА закінчує свою роботу після скінченного числа кроків.

Доведення. Дійсно, за умовою теореми і згідно з теоремою 12.2.3 алгоритм МВА в кожному стані U - Y -програми буде спадну послідовність нормальних конгруентностей $N_a^{(0)} \supseteq \supseteq N_a^{(1)} \supseteq \dots \supseteq N_a^{(n)} \supseteq \dots$, яка скінчена за умовою обриву таких ланцюгів.

Теорема 12.2.5. Якщо в алгебрі $T(\Omega, R)$ виконується умова обриву зростаючих ланцюгів нормальних конгруентностей, N_0 — нормальна конгруентність, і перетин нормальних конгруентностей є нормальнюю конгруентністю, то алгоритм МНА завершує свою роботу після скінченного числа кроків.

Доведення. Дійсно, за умовою теореми і згідно з теоремою 12.2.3 алгоритм МНА в кожному стані U - Y -програми буде зростаючу послідовність нормальних конгруентностей $N_a^{(0)} \subseteq \subseteq N_a^{(1)} \subseteq \dots$, яка скінчена за умовою обриву зростаючих ланцюгів.

Зауважимо, що умови обриву зростаючих і спадаючих ланцюгів нормальних конгруентностей виконуються для спадково-вільних алгебр, тобто таких алгебр, всі підалгебри яких вільні в одному і тому ж класі. До спадково-вільних алгебр належать такі важливі класи алгебр, як абсолютно-вільні алгебри, векторні простори, вільні групи, вільні абелеві групи тощо.

Наслідок 12.2.1. Якщо алгебра $T(\Omega, R)$ спадково-вільна, то алгоритм МНА завершує свою роботу за скінченне число кроків [22].

Теорема 12.2.6. Якщо алгоритм МВА завершує свою роботу після скінченного числа кроків і множини співвідношень, які при цьому одержуються в кожному стані U - Y -програми в процесі його роботи, алгебраїчно замкнуті, то результат роботи алгоритму МВА не залежить від способу обходу станів U - Y -програми (і від вибору початкової сукупності простих шляхів), а множина інваріантів N для всякого стану $a \in A$ збігається з множиною

$$\bigcap_{l = (a_0, a)} \text{ef}(N_0, y_l).$$

Доведення теореми ґрунтуються на таких лемах.

Лема 12.2.1. Якщо функція ef дистрибутивна і алгоритм МВА завершує свою роботу після скінченного числа кроків, то

$$(\forall a \in A) (N_a = \bigcap_{(a', u, y, a) \in S} \text{ef}(N_0, y)).$$

Доведення виконується індукцією за числом попадань в стан a .

$m = 2$ (база). За алгоритмом маємо

$$N_a = N_a \cap \left(\bigcap_{(a', u, y, a) \in S} \text{ef}(N_{a'}, y) \right)$$

для деякого $i > 1$. Але

$$N_a^{(2)} = \text{ef}(N_{a'}^{(1)}, y) \cap \left(\bigcap_{(a', u, y, a) \in S} \text{ef}(N_{a'}^{(i)}, y) \right),$$

і оскільки $N_{a'}^{(1)} \supseteq N_{a'}^{(i)}$ для $i > 1$, то внаслідок монотонності дистрибутивної функції ef можна записати

$$N_a^{(2)} = \bigcap_{(a', u, y, a) \in S} \text{ef}(N_{a'}^{(i)}, y).$$

$m > 2$ (кrok індукції). Нехай рівність $N_a^{(k)} = \bigcap_{(a', u, y, a) \in S} \text{ef}(N_{a'}^{(i)}, y)$

справедлива для всіх $k < m$. Тоді за алгоритмом

$$\begin{aligned} N_a^{(m)} &= N_a^{(m-1)} \cap \left(\bigcap_{(a', u, y, a) \in S} \text{ef}(N_{a'}^{(i)}, y) \right) = \\ &= \left(\bigcap_{(a', u, y, a) \in S} \text{ef}(N_{a'}^{(j)}, y) \right) \cap \left(\bigcap_{(a', u, y, a) \in S} \text{ef}(N_{a'}^{(i)}, y) \right) \end{aligned}$$

($i > j$) за припущенням індукції. На основі монотонності ef маємо

$$N_a^{(m)} = \left(\bigcap_{(a', u, y, a) \in S} \text{ef}(N_{a'}^{(j)}, y) \right) \cap \left(\bigcap_{(a', u, y, a) \in S} \text{ef}(N_{a'}^{(i)}, y) \right) = \\ = \bigcap_{(a', u, y, a) \in S} \text{ef}(N_{a'}^{(i)}, y).$$

Тепер твердження леми випливає з умови завершенності алгоритму МВА.

Лема 12.2.2. Якщо функція ef дистрибутивна, і алгоритм МВА завершує свою роботу після скінченного числа кроків, то

$$(\forall a \in A) \left(N_a = \bigcap_{l = l(a_0, a)} \text{ef}(N_0, y_l) \right).$$

Доведення проводиться індукцією за довжиною $|l|$ шляху $l = l(a_0, a)$.

$|l| = 0$. Тоді $a = a_0$ і $N_{a_0} = N_0 = N$, оскільки N_0 далі не змінюється, то база індукції має місце.

$|l| > 0$. Припустимо, що для всіх $|l'| < |l|$ твердження леми справедливе, коли $a \neq a_0$. Тоді за лемою 12.2.1 маємо

$$N_a = \bigcap_{(a', u, y, a) \in S} \text{ef}(N_{a'}, y).$$

За припущенням індукції і внаслідок дистрибутивності функції ef одержуємо

$$N_a = \bigcap_{(a', u, y, a) \in S} \text{ef} \left(\bigcap_{l' = l(a_0, a')} \text{ef}(N_{a'}, y_{l'}) , y \right) = \bigcap_{l = l(a_0, a)} \text{ef}(N_0, y_l).$$

Лема доведена.

Доведення теореми 12.2.6 тепер очевидним чином випливає з теореми 12.2.4 та лем 12.2.1 і 12.2.2.

Якщо множина інваріантів будь-якого стану U - Y -програми збігається з множиною

$$\bigcap_{l = l(a_0, a)} \text{ef}(N_0, y_l),$$

то вона буде називатися **повною відносно генератора** інваріантів, мови L і початкової множини N_0 .

Теорема 12.2.6 дає умови, при яких алгоритм МВА генерує повні множини співвідношень відносно початкової сукупності співвідношень. Коли функція ef не є дистрибутивною, то одержання повної системи співвідношень стає алгоритмічно нерозв'язною задачею. Це випливає з такого твердження.

Теорема 12.2.7. Якщо функція ef не дистрибутивна, то можна побудувати U - Y -програму A з початковою множиною співвідношень

N_0 , для якої не існує алгоритму, що генерує повну систему інваріантів відносно N_0 для цієї U - Y -програми.

Доведення теореми виконується шляхом зведення даної проблеми до модифікованої проблеми відповідностей Поста (МПВП) [81], яка формулюється таким чином.

Дано два слова $p = x_1x_2\dots x_n$ і $p' = y_1y_2\dots y_n$ в алфавіті $\{0, 1\}$. Потрібно знайти послідовність цілих чисел i_1, i_2, \dots, i_k , таку, що має місце рівність $x_{i_1}x_{i_2}\dots x_{i_k} = y_{i_1}y_{i_2}\dots y_{i_k}$. Добре відомо, що проблема Поста алгоритмічно нерозв'язна.

Розглянемо U - Y -програму на рис. 12.2.1.

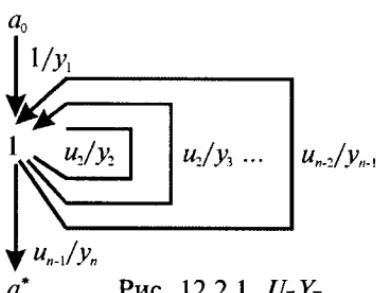


Рис. 12.2.1. U - Y -програма

Пов'яжемо з цією програмою гратку з такими елементами:

(1) 0 — нульовий елемент структури;

(2) \$ — спеціальний елемент, за допомогою якого фіксується відсутність розв'язку МПВП;

(3) всі рядки цілих чисел від 1 до k , які починаються з 1.

Операція перетину на структурі визначається як $x \wedge y = 0$ тоді і

тільки тоді, коли $x \neq y$. Значить, якщо $x = y$, то $x = y$ або $x = 0$.

Значення функції $ef(M, y_i)$, $i = 1, 2, \dots, k + 1$, задаються так:

(а) $ef(\emptyset, \epsilon) = e$ — тотожна функція, а ϵ — тотожний оператор присвоювання;

(б) для всіх $i = 1, 2, \dots, k$ маємо:

(б1) якщо $M = \{s\}$ — рядок цілих чисел, що починається з 1, то $ef(M, y_i) = s_i$;

(б2) $ef(\emptyset, y_i) = 0$;

(б3) $ef($, y_i) = $$;

(в) $ef(M, y_{i+1}) = \begin{cases} 0, & \text{якщо } s - \text{розв'язок МПВП;} \\ \$ & \text{— в іншому випадку;} \end{cases}$

(в1) $ef(\emptyset, y_{k+1}) = 0$;

(в2) $ef($, y_{k+1}) = $$;

(г) для будь-якого $M = \{s\}$ $ef(M, y_1) = \{1\}$ — рядок, що складається з одного елемента 1.

Лема 12.2.3. Структура, означення (1)–(3) якої наведено вище, є нижньою напівструктурою, а функція ef монотонна.

Доведення. З означення операції перетину очевидним чином випливає, що вона ідемпотентна, комутативна і асоціативна, тобто задає деяку напівструктуру, а довжини всіх строго спадних ланцюгів обмежені числом 2.

Залишається показати монотонність функції ef .

Нехай $M = \{s\} < M' = \{s'\}$. Тоді:

а) для довільних M, M' $\text{ef}(M, y_i) = \text{ef}(M', y_i) = \{1\}$ (внаслідок (г));

б) якщо $M < M'$, то $M = M'$ або $M = \emptyset$, і тому $\text{ef}(M, y_i) < \text{ef}(M', y_i)$ випливає для всіх $i = 2, 3, \dots, k + 1$.

Лема доведена.

Доведення теореми 12.2.7 тепер випливає з таких міркувань: якби для заданої U - Y -програми існував алгоритм побудови повної системи інваріантів, то він би давав розв'язок МПВП, а це суперечило б алгоритмічній нерозв'язності МПВП.

Виникає природне питання, на яку множину інваріантів можна розраховувати в цьому випадку, і які властивості цих множин? Відповідь на ці питання дає така теорема.

Теорема 12.2.8. Якщо алгоритм МВА завершує роботу через скінченне число кроків, і функція ef не є дистрибутивною, то побудована алгоритмом множина інваріантів N для будь-якого стану a із A є максимально фіксованою точкою розв'язку системи рівнянь

$$\begin{cases} X_a = X_a \cap_{(a', u, y, a) \in S} \text{ef}(X_{a'}, y), \\ X_{a_0} = N_0. \end{cases} \quad (12.2.1)$$

Д о в е д е н н я. Очевидно, що після завершення роботи алгоритму МВА множина інваріантів N_a в довільному стані $a \in A$ задовольняє систему (12.2.1). Нехай K_a — довільний інший розв'язок системи (12.2.1). Покажемо індукцією за числом попадань в стан a , що після m -го попадання $(\forall a \in A)(K_a \subseteq N_a)$, де N_a — множина співвідношень у стані a , одержана за допомогою алгоритму МВА.

$m = 0$. При $m = 0$ маємо $K_{a_0} = N_a^{(0)} = N_0$ і $K_a = 1$, де 1 — одиниця структури L (з метою оптимізації цей крок в алгоритмі МВА пропущений, оскільки $(\forall a \in A)(N_a \in L, L = 1 \text{ і } N_a \cap 1 = N_a)$).

Отже, при $m = 0$ твердження теореми справедливе.

$m > 0$. Припустимо, що $K_a \subseteq N_a^{(m-1)}$ для довільного $a \in A$.

Покажемо, що $K_a \subseteq N_a^{(m)}$. Дійсно,

$$\begin{aligned} N_a^{(m)} &= N_a^{(m-1)} \cap \left(\bigcap_{(a', u, y, a) \in S} \text{ef}(N_{a'}^{(m-1)}, y) \right) \supseteq K_a \cap \\ &\cap \left(\bigcap_{(a', u, y, a) \in S} \text{ef}(K_{a'}, y) \right) = K_a \end{aligned}$$

за припущенням індукції, монотонності функції ef і того, що K_a — розв'язок (12.2.1). Для решти станів a' маємо $N_a^{(m)} = N_{a'}^{(m-1)}$. Таким чином, $K_a \subseteq N_a^{(m)}$. Тепер твердження теореми випливає з умови завершення алгоритму МВА.

Теорема доведена.

Наслідок 12.2.2. Результат роботи алгоритму МВА не залежить від порядку вибору елементів із множини C .

§ 12.3. ПОШУК ІНВАРІАНТІВ У ПРОГРАМАХ НАД ВІЛЬНИМИ АЛГЕБРАМИ ДАНИХ

Перейдемо до розгляду U - Y -програм над конкретними алгебрами даних.

1. АБСОЛЮТНО ВІЛЬНІ АЛГЕБРИ

Розглянемо випадок, коли алгебра даних U - Y -програми абсолютно вільна, тобто D збігається з $T(\Omega, R)$.

Розв'язок задачі про співвідношення. Нехай (a, u, y, b) — деякий перехід в U - Y -програмі A , де

$$y = (r := t(r)) = (r_1 := t_1(r), \dots, r_m := t_m(r)),$$

$r = (r_1, \dots, r_m)$, $t_i(r) \in T(\Omega, R)$, $i = 1, 2, \dots, m$, і $t[t_1 \leftarrow t_2]$ означає підстановку терма t_2 в терм t замість деякого входження терма t_1 . Будемо також вважати, що елементи $r_i := t_i(r)$ в операторі y упорядковані за зростанням довжини термів $t_i(r)$. Вияснимо структуру множини $\text{ef}(\emptyset, y)$.

Оскільки алгебра даних абсолютно вільна, то $t(r) = t'(r)$ має місце тоді і тільки тоді, коли терми $t(t_1, \dots, t_m)$ і $t'(t_1, \dots, t_m)$ графічно збігаються. Виділимо в $\text{ef}(\emptyset, y)$ елементи, що мають вигляд $r_i = t(r_1, \dots, r_{i-1})$, які будемо називати *простими*. Якщо d — простий елемент, то $LT(d)$ і $RT(d)$ — відповідно ліва і права частини d . Введемо також позначення $RL(r_i) = \{d \in \text{ef}(\emptyset, y) \mid LT(d) = r_i\}$ і

$$RL = \bigcup_{i=1}^m RL(r_i).$$

Нехай $d, d_1 \in RL$. Застосувати співвідношення d_1 до d означає перейти від d до елемента d' , що має вигляд $LT(d) = RT(d)[LT(d_1) \leftarrow RT(d_1)]$. При цьому будемо говорити, що d' — безпосередній наслідок d ($d \mid= d'$). Транзитивне замикання відношення безпосереднього слідування будемо називати просто *слідуванням*, а елемент d'' , одержаний внаслідок слідування з елемента d , — *наслідком* ($d \mid= d''$). Відношення $\mid=$ буде, очевидно,

відношенням часткового порядку на RL . З означення простого елемента випливає, що $RL(r_i)$ і RL скінченні.

Лема 12.3.1. $RL(r_i)$ має єдиний мінімальний (відносно відношення \models) елемент.

Доведення. Припустимо, що в $RL(r_i)$ існує хоча б два мінімальні елементи d і d' . Представимо (для зручності подальших викладок) $RT(d)$ і $RT(d')$ у вигляді складених об'єктів, графи яких є деревами, і виконаємо паралельний обхід їх вершин зліва направо, зверху вниз, порівнюючи відмітки цих вершин. У цьому разі можуть виникнути такі ситуації:

- 1) відмітки вершин однакові;
- 2) $r' \models r$ ($r \models r'$), $r, r' \in R$;
- 3) $r \models \omega$ ($\omega \models r$), $r \in R$, $\omega \in W$;
- 4) $\omega \models \omega'$ ($\omega' \models \omega$), $\omega, \omega' \in \Omega$;

Тоді залежно від ситуації виконуємо такі дії.

1. Переходимо до наступних вершин згідно з правилами обходу.

2. Виконуємо підстановки замість всіх r_k ($k = 1, 2, \dots, i - 1$), які входять в $RT(d)$ і $RT(d')$, відповідних термів t_k . Оскільки $d, d' \in RL(r_i) = ef(\emptyset, y)$, то одержані терми повинні графічно збігатися. Значить, в RL існує елемент $r = r'$ ($r' = r$), за допомогою якого із d (d') одержуємо d' (d). Суперечність із мінімальністю d, d' .

3. Доведення аналогічне 2.

4. Приходимо до суперечності з тим, що $d, d' \in RL(r_i) = ef(\emptyset, y)$, оскільки при заміні всіх r_k термами $t_k(r)$, $k = 1, 2, \dots, i - 1$, графічного збігу термів $RT(d)$ і $RT(d')$ не може бути.

Лема доведена.

Нехай P — множина мінімальних елементів із RL .

Лема 12.3.2. $C(P) = ef(\emptyset, y)$, $|P| \leq m = |R|$.

Доведення. Нерівність $|P| \leq m = |R|$ випливає з попередньої леми.

Нехай d — довільний елемент із $ef(\emptyset, y)$, що має вигляд $t(r) = t'(r)$. Представимо терми t, t' так, як це було в лемі 12.3.1, складеними об'єктами у вигляді дерев і виконаємо паралельний обхід їх вершин зліва направо, зверху вниз. Знову можливі розглянуті вище чотири випадки, які обумовлені різними комбінаціями відміток вершин, які розглядаються в даний момент. Залежно від ситуації:

- 1) продовжуємо обхід дерев згідно з правилами обходу;
- 2) і 3) повинні існувати відповідні прості елементи $r_i = r_j$ ($r_j = r_i$) і $r_i = t(r_1, \dots, r_{i-1})$. Ці елементи будемо називати **розділом спiввiдношення d на простi**. Застосовуючи операції з Ω в тому

порядку, в якому вони вказані в термах t , t' до елементів розкладу d , побудуємо d . Внаслідок довільності d і того, що прості елементи лежать в $C(P)$, одержуємо включення $C(P) \supseteq \text{ef}(\emptyset, y)$. З іншого боку, оскільки $P \subseteq \text{ef}(\emptyset, y)$, то $C(P) \subseteq \text{ef}(\emptyset, y)$.

Що і потрібно було довести.

Наслідок 12. 3.1. $C(P)$ є нормальнюю конгруентністю.

Дійсно, $C(P) = \text{ef}(\emptyset, y)$, де $y = (r_1 := t_1(r_1, \dots, r_m), \dots, r_i := t_i(r_1, \dots, r_m))$, а $r_i = t_i(r_1, \dots, r_m)$ — прості елементи із $C(P)$.

Розв'язок задачі про перетин. Нехай $d_1, d_2 \in \text{ef}(\emptyset, y) = C(P)$. Позначимо $d_1(P) \models d_2$ той факт, що d_2 — наслідок d_1 в базисі P , тобто що d_2 одержаний з d_1 шляхом застосування елементів з P . Нехай P_1, P_2 — алгебраїчні базиси деяких множин співвідношень і $d \in P_1, d' \in P_2$, такі, що $LT(d) = LT(d')$ і $C(P_1, d) = \{d \mid d(P_1) \models \models d\}$, $C(P_2, d') = \{d' \mid d'(P_2) \models \models d'\}$. Елементи множини $T = C(P_1, d) \cap C(P_2, d')$ будемо називати *спільними кратними* d і d' відносно пари (P_1, P_2) , а мінімальний елемент цієї множини позначимо $\text{HCK}(d, d')/(P_1, P_2)$, коли $T \neq \emptyset$. Існування мінімального елемента випливає із скінченності множини T , а єдиність — з наведено-го нижче процесу його побудови (процес побудови НСК).

У процесі побудови НСК за елементами d і d' будеться третій елемент d'' таким чином. Спочатку в P_1 і P_2 виділяються сукупності рівностей, що мають вигляд $r_{i1} = r_{i2}, r_{i3} = r_{i2}, \dots, r_{ip} = r_{ip-1}$. Поставимо у відповідність кожній змінній r_{ij} одне і те ж число q , яке називається *індексом* ($\text{ІНД}(r_{ij})$), $j = 1, 2, \dots, p$. Якщо змінна r_{ij} уже одержала раніше індекс, то всі останні змінні цієї сукупності одержують індекс змінної r_{ij} . Розкладавши індекси всім таким сукупностям змінних, виконаємо паралельний лівий обхід зверху вниз дерев $RT(d)$ і $RT(d')$, порівнюючи відмітки їх вершин. Залежно від ситуації, яка може виникнути під час обходу (див. доказання леми 12.3.1), виконуємо такі дії.

1. Копіюємо вершину і переходимо до порівняння наступних вершин згідно з порядком обходу.

2. Якщо $\text{ІНД}(r_i) = \text{ІНД}(r_j)$ виконується тільки в P_1 , тобто $r_i = r_j$ лежить в P_1 , але не в P_2 , то копіюємо вершину з відміткою r_i і переходимо до порівняння наступних вершин.

Якщо $\text{ІНД}(r_i) = \text{ІНД}(r_j)$ виконується в P_2 , то копіюємо вершину з відміткою r_i і переходимо до порівняння наступних вершин. (Тут не вимагається, щоб рівність $\text{ІНД}(r_i) = \text{ІНД}(r_j)$ виконувалась тільки в P_2 , бо якщо вона виконується і в P_1 , і в P_2 , то копіювати можна будь-яку з вершин: r_i або r_j .)

3. Знайти $d_1 \in P_1$ ($d_2 \in P_2$), такий, що $LT(d_1) = r_j$ ($LT(d_2) = r_i$). Якщо d_1 (d_2) існує, то застосувати d_1 (d_2) до даного входження r_j

(r_i) в $RT(d)$ ($RT(d')$) і перейти до порівняння наступних вершин. У протилежному випадку перейти на крок 4.

4. Витерти побудоване дерево і закінчити процес порівняння термів.

Процес успішно закінчується, якщо він завершився обходом термів $RT(d)$ і $RT(d')$ без попадання на крок 4. У цьому випадку процес називається застосовним (тобто таким, що може бути застосований) до $RT(d)$ і $RT(d')$. Якщо процес застосовний і t — побудований терм, то складемо рівність d'' , що має вигляд $LT(d) = t$.

Лема 12. 3.3. $d'' \in HCK(d, d')/(P_1, P_2)$.

Доведення. Припустимо, що $d'' \notin HCK(d, d')/(P_1, P_2)$. Тоді існує $d_1 \in HCK(d, d')/(P_1, P_2)$, менший d'' , і хоча б один $c \in T = C(P_1, d) \cap C(P_2, d')$, такий, що d_1 одержується з d'' застосуванням c . Нехай $Q = (d'_1, \dots, d'_k)$ і $Q' = (d''_1, \dots, d''_k)$ — послідовності елементів, які застосовувались до $RT(d)$ із P_1 і до $RT(d')$ із P_2 відповідно в процесі побудови $RT(d'')$. Оскільки $RT(c)$ — підтерм терма $RT(d)$, то із Q і Q' вилучимо ті елементи, які застосовувались до $RT(d)$ і $RT(d')$ в процесі побудови даного входження $RT(c)$ в $RT(d'')$. Застосуємо до $RT(d)$ і $RT(d')$ процес побудови HCK з новими послідовностями Q і Q' . В результаті одержимо елемент \bar{d} , менший d'' , і \bar{d} одержується із d'' застосуванням c . Це значить, що при побудові $RT(d'')$ на деякому кроці до вершин з однаковими відмітками були застосовані співвідношення c_1 із P_1 і c_2 із P_2 , такі, що $LT(c_1) = LT(c_2) = LT(c)$. Одержуємо суперечність з процесом побудови $RT(d'')$.

Лема доведена.

Наслідок 12.3.2. $HCK(d, d')/(P_1, P_2) = HCK(d', d)/(P_1, P_2)$.

Справедливість цього наслідку випливає із способу побудови HCK .

Видіlimо пари елементів (d, d') , $d \in P_1$, $d' \in P_2$, такі, що $LT(d) = LT(d')$. Для кожної такої пари побудуємо $HCK(d, d')/(P_1, P_2)$, і нехай PT — множина побудованих таким чином елементів.

Лема 12.3.4. $C(PT) = C(P_1) \cap C(P_2)$, $|PT| \leq \min(|P_1|, |P_2|)$.

Доведення. Справедливість нерівності $|PT| \leq \min(|P_1|, |P_2|)$ випливає із способу побудови PT .

Нехай d — довільний елемент із $T' = C(P_1) \cap C(P_2)$. Розкладемо d на прості елементи, і нехай b — деякий елемент цього розкладу (див. доведення леми 12.3.2). Якщо $b \in C(PT)$, то все доведено. Припустимо, що $b \notin PT$. Тоді існує c із PT , такий, що $LT(b) = LT(c)$ (інакше має місце суперечність з тим, що $d \in T'$).

Нехай у процесі побудови b із c одержані послідовності елементів Q і Q' , які застосувались до c . Перші елементи цих послідовностей d_1 і d_2 мають однакові ліві частини. Нехай Q_1 і Q'_1 – послідовності елементів із P_1 і P_2 відповідно, які застосувались у процесі побудови $\text{HCK}(d_1, d_2)/(P_1, P_2)$ (процес побудови HCK застосовний до d_1, d_2 , оскільки в протилежному випадку маємо суперечність з тим, що $c(P_1) \models b$ або $c(P_2) \models b$). Вилучимо із Q і Q' їх підпослідовності Q_1 і Q'_1 . В частинах Q і Q' , які залишилися, перші елементи знову будуть мати рівні ліві частини. Виконуючи над ними аналогічну процедуру і повторюючи її доти, доки Q і Q' не стануть пустими, одержимо послідовність $Q = (d_1, \dots, d_k)$, $k = \min(|Q|, |Q'|)$, елементи якої за побудовою належать до PT . Значить, $b \in C(PT)$, і тому $d \in C(PT)$ і $C(PT) \supseteq T'$. З іншого боку, очевидно, що $PT \subseteq T'$, отже, $C(PT) \subseteq T'$, звідки випливає рівність $C(PT) = C(P_1) \cap C(P_2)$.

Лема доведена.

Наслідок 12.3.3. $C(PT)$ є нормальнюю конгруентністю.

Дійсно, $C(PT) = \text{ef}(\emptyset, y)$, де $y = (r_1 := t_1(r), \dots, r_m := t_m(r))$, в якому $t_i(r) = r_i$, якщо не існує $d \in PT$, такого, що $LT(d) = r_i$, інакше $t_i(r) = RT(d)$, $LT(d) = r_i$, $i = 1, 2, \dots, m$.

Наслідок 12.3.4. Якщо $C(PT) = C(P_1) \cap C(P_2) \cap \dots \cap C(P_k)$, то побудова PT не залежить від порядку, в якому перелічені члени P_1, P_2, \dots, P_k .

Справедливість цього випливає з наслідку 12.3.4 і способу побудови PT .

Розглянемо тепер алгоритми розв'язку основних задач і з'ясуємо їх ефективність.

Алгоритм побудови базису множини $\text{ef}(M, y)$. Спочатку розглянемо розв'язок задачі суперпозиції операторів лінійної ланки, яка виникає при побудові базису множини $\text{ef}(M, y)$. За лемою 12.3.3 і наслідком 12.3.1 ця задача зводиться до побудови базису $\text{ef}(\emptyset, y_2)$, де $y_2 = y_1 * y$ представляє оператор, який є суперпозицією операторів присвоювання y_1 і y , а конкатенація цих операторів буде називатися далі **розділенням** оператором присвоювання. Опишемо насамперед алгоритм суперпозиції операторів присвоювання.

Припустимо, що змінні r_1, \dots, r_m із R перенумеровані (закодовані) цілими числами з проміжку від N до $N + m$, розширений оператор присвоювання $y = (r_1 := t_1(r), \dots, r_m := t_m(r))$ представлений складеним об'єктом U (рис. 12.3.1), вершини якого перенумеровані цілими числами від K до $K + n$, $*$ означає пустий складений.

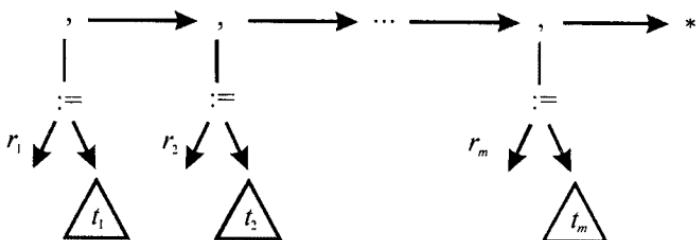


Рис. 12.3.1. Складений об'єкт, що представляє розширеній оператор присвоювання

Алгоритм виконується за один обхід складеного U за допомогою одновимірного масиву T довжини $m = |R|$. В масиві T зберігаються номери початкових вершин складених, які представляють праві частини операторів присвоювання. Таким чином, якщо $r_i := t_i(r)$ — оператор присвоювання і змінна r_i має номер q , то в $T(q)$ зберігається номер початкової вершини складеного $t_i(r)$.

Робота алгоритму зводиться до таких дій: якщо в процесі обходу деякої правої частини $t_i(r)$ покажчик оглядає вершину з відміткою $r_j \in R$, то за номером q змінної r_j із масиву $T(q)$ дістается відповідний номер початкової вершини правої частини оператора присвоювання, який передує даному, і який останнім змінював значення змінної r_j ; посилання на вершину, яка оглядається, замінюється посиланням на вершину з номером $T(q)$.

Після закінчення обходу складеного t_j модифікується елемент $T(p)$, де p — номер змінної r_j .

КОМПОЗИЦІЯ (U, m)

Вихід: складений U , який представляє розширеній оператор присвоювання y_1y_2 , і число елементів у множині R .

Вихід: складений, який представляє оператор y_2 , тобто композицію початкових операторів присвоювання.

Пам'ять: масив T , що складається з m елементів, показчики X, Y .

Метод: $X \rightarrow U; T := 0;$

$T(\text{номер}(\arg(X, 1, 1))) := \text{номер}(\arg(X, 1, 2));$

поки $\arg(X, 2) \neq *$ **виконувати;**

$X \rightarrow \arg(X, 2); Y \rightarrow \arg(X, 1, 2);$

в процесі обходу складеного $\arg(X, 1, 2)$ **виконати;**

якщо позначка ($Y = r_i \& T(\text{номер}(r_i)) \neq 0$ **то**

замінити посилання на вершину Y посиланням на вершину $T(\text{номер}(r_i))$;

кінець обходу;

$T(\text{номер}(\arg(X, 1, 1))) := T(\text{номер}(\arg(X, 1, 2)));$

кц;

$X \rightarrow U;$
поки $\arg(X, 2) \neq *$ **виконувати;**
якщо $T(\text{номер}(\arg(X, 1, 1))) \neq \text{номер}(\arg(X, 1, 2))$ **то**
 $(\arg(X, 1, 1) := *; \arg(X, 1) := *; X := \arg(X, 2))$
інакше $X \rightarrow \arg(X, 2);$
кц;

Лема 12.3.5. Часова складність алгоритму КОМПОЗИЦІЯ пропорційна числу вершин складеного U .

Доведення очевидне.

Приступимо тепер до опису алгоритму побудови множини співвідношень $\text{ef}(P, y)$, де P — алгебраїчний базис множини M .

У цьому алгоритмі використовуються алгоритми, які описуватися тут не будуть, оскільки один із них описаний в розділі 5, а інші досить відомі — це алгоритми ПРИВЕСТИ ЗАГАЛЬНІ ПІДВИРАЗИ і УПОРЯДКУВАТИ, які виконують відповідно склеювання загальних підвиразів у складеному, що представляє оператор y_2 , і упорядкування елементів оператора y_2 за довжиною термів, що стоять в правих частинах операторів присвоювання.

СПІВВІДНОШЕННЯ (U, m)

Вхід: складений U , який представляє оператори y_1y , і $m = |R|$.

Вихід: складений, який представляє базис множини $\text{ef}(\emptyset, y_2)$;

Пам'ять: масив $T[1:m]$ і показчики Z, V .

Метод: $Z \rightarrow U$;

КОМПОЗИЦІЯ (U, m);

$Z \rightarrow U$;

УПОРЯДКУВАТИ (Z);

$Z \rightarrow U$;

ПРИВЕСТИ ЗАГАЛЬНІ ПІДВИР (Z);

$Z \rightarrow U; /*$ заповнюємо масив $T */$

поки $\arg(Z, 2) \neq *$ **виконувати;**

$T(\text{номер}(\arg(Z, 1, 2))) := \text{номер}(\arg(Z, 1, 1));$

кц;

$Z \rightarrow U; /*$ виконуємо обернені заміни */

в процесі обходу Z виконати;

якщо $T(\text{номер}(Z)) \neq 0$ то замінити посилання на вершину Z посиланням на вершину

$T(\text{номер}(\text{відмітка}(Z)));$

кя;

кінець обходу;

$Z \rightarrow U; T := 0; /*$ мітимо вершини */

доки $\arg(Z, 2) \neq *$ **виконати;** $T(\text{номер}(\arg(Z, 1, 1))) := 1;$

кц;
 $Z \rightarrow U$; /* виділяємо співвідношення */
 доки $Z \neq *$ виконувати;
 $V \rightarrow \arg(Z, 1, 2)$;
 в процесі обходу V виконати;
 якщо V первинний і $T(\text{номер}(V)) = 1$ то
 перервати обхід;
 кя;
 кінець обходу;
 $T(\text{номер}(Z)) := 0$;
 якщо обхід закінчено без переривань то
 $Z \rightarrow \arg(Z, 2)$ інакше $Z := \arg(Z, 2)$
 кя;
 кц;
 кінець.

Лема 12.3.6. Часова складність алгоритму СПІВВІДНОШЕННЯ пропорційна величині n , де n — число вершин у початковому складеному.

Доведення. Перший крок алгоритму за лемою 12.3.5 має лінійну оцінку часової складності. Алгоритми УПОРЯДКУВАТИ і ПРИВЕСТИ ЗАГАЛЬНІ ПІДВИР, як ми знаємо з розділу 10, теж виконуються за час, лінійно залежний від числа n . Решта кроків алгоритму виконуються, очевидно, за один обхід початкового складеного.

Лема доведена.

Алгоритм розв'язання задачі про перетин ПЕРЕТИН (P_1, P_2, m)

Вхід: складені U, V , які представляють відповідно базиси P_1, P_2 .

Вихід: складений W , який представляє базис PT множини співвідношень $C(P_1) \cap C(P_2)$.

Пам'ять: масив $T[1:m]$, покажчики $X, X1, Y, Z$.

Метод: $X \rightarrow U; Y \rightarrow V; W := *; Z \rightarrow W;$

$T := 0$;

доки $\arg(Y, 2) \neq *$ виконати;

 якщо $T(\text{номер}(\arg(Y, 2))) \neq 0$ то

$Y := T(\text{номер}(\arg(Y, 2)))$;

$X1 := \arg(Y, 1, 2)$;

 в процесі паралельного обходу $X1, Y$ виконати;

 якщо позначення($X1$) \neq позначення(Y) то

 /* якщо номер(Y) & номер($X1$) лежать в

```

одному класі еквівалентності то */  

застосувати співвідношення ( $X1, Y$ );  

/* інакше */  

якщо співвідношення не застосовні то  

    закінчти аварійно обхід  $X1, Y$ ;  

     $X1 := *; Y := *$ ;  

    кя;  

    кя;  

кінець обходу;  

якщо обхід закінчився не аварійно то  

     $Z := * = *$ ; /* заготовка для співвідношення */  

     $\arg(Z, 1, 2) := X1$ ;  

     $\arg(Z, 1, 1) := \arg(X, 1, 1)$ ;  

    кя;  

 $X \rightarrow \arg(X, 2)$ ;  

кц;

```

Лема 12.3.7. Часова складність алгоритму ПЕРЕТИН пропорційна величині $O(m * (n_1 + n_2))$, де $m = |R|$, n_i — число вершин у складеному, який представляє базис P_i , $i = 1, 2$.

Д о в е д е н н я. Насамперед покажемо, що число вершин у складеному, який представляє базис PT , не перевищує величини $n_1 + n_2$.

Нехай d_1 — перший елемент базису P_1 . Знайдемо входження $RT(d_1)$ в складеному P_2 . Якщо таке входження є, то побудуємо клас еквівалентності $A_1 = \{i, j, i', j'\}$, де $i = \text{номер}(LT(d_1))$, $j = \text{номер}(RT(d_1))$, $i' = \text{номер}(LT(d'))$, $j' = \text{номер}(RT(d'))$, якщо d' в P_2 існує. Якщо ж d' не існує, то $A_1 = \{i, j, j'\}$, де j' — початкова вершина складеного, який представляє підтерм $RT(d_1)$ в P_2 . Після цього візьмемо наступний елемент d_2 із P_1 і виконаємо над ним ту саму процедуру, що й над d_1 , але з урахуванням класу A_1 , тобто знаходимо всі підтерми в P_2 , які збігаються з $RT(d_2)$ з точністю до відміток вершин із A_1 . Із початкових вершин складених, що представляють відповідні підтерми в P_1 і P_2 , будуємо клас A_2 і т.д. Повторюємо це доти, доки не вичерпаемо всіх елементів із P_1 .

Якщо ж на першому етапі наведеної вище процедури $RT(d_1)$ не входить в P_2 як підтерм, то беремо перший елемент d'_1 із P_2 і виконуємо над ним те саме, що і над d_1 , будуючи клас еквівалентності A'_1 . Якщо ж і $RT(d')$ не входить в P_1 як підтерм, то беремо другий елемент d_2 із P_1 і все повторюємо заново.

В результаті побудуємо класи еквівалентності A_1, A_2, \dots, A_m (A'_1, A'_2, \dots, A'_m) (m, m') вершин, що є початковими вершинами складених, за якими і будеться базис PT . Із процедури побудови

класів еквівалентності очевидним чином випливає, що число вершин у складеному PT обмежене величиною $n_1 + n_2$.

Маючи в своєму розпорядженні класи еквівалентності A_i (A'_i), побудову складеного PT алгоритмом ПЕРЕТИН можна виконати за час, пропорційний величині $O(n_1 + n_2)$. Для цього необхідно лише вставити перевірку належності вершин, які розглядаються в даний момент (до яких необхідно застосовувати співвідношення), до одного і того ж класу еквівалентності. В алгоритмі ця перевірка подана як коментар.

Описана процедура побудови класів A_i (A'_i) відповідає m' -кратному обчисленню конгруентного замикання на ациклических (бінарних) складених, і згідно з результатами розділу 6 часова складність цієї процедури пропорційна величині

$$O(m'(n_1 + n_2)) \leq O(m(n_1 + n_2)).$$

Таким чином, часова складність алгоритму ПЕРЕТИН обмежена величиною $O(m(n_1 + n_2))$, що і потрібно було показати.

Теорема 12.3.1. Алгоритм МВА (МНА) завершує свою роботу після скінченного числа кроків. Число кроків алгоритму МВА обмежене величиною $\text{const} * (m * n)^2$, де $m = |R|$, n — число вершин у складеному, що представляє U - Y -програму.

Д о в е д е н н я. Закінчення роботи алгоритму МВА (МНА) випливає з наслідку 12.3.3, теорем 12.2.4 і 12.2.6, оскільки в абсолютно вільній алгебрі виконується умова обриву спадних (зростаючих) послідовностей нормальних конгруенцій. Довжина максимального строго спадного ланцюга обмежена величиною m , а побудова базису $\text{ef}(N, y)$ і базису перетину (хоча число вершин і зростає, але воно обмежене зверху числом вершин у складеному, який представляє U - Y -схему програми) обмежені відповідно величинами n і $m * n$.

Таким чином, часова складність алгоритму МВА обмежена величиною $O((m * n)^2)$, де $m = |R|$, n — розмір U - Y -схеми програми.

§ 12.4. ВЕКТОРНІ ПРОСТОРИ

1. РОЗВ'ЯЗКИ ОСНОВНИХ ЗАДАЧ

Нехай $T_D(R)$ — векторний афінний простір над деяким полем P . Тоді всякий оператор присвоювання $y \in Y$ можна записати так:

$$\begin{aligned} r_1 &:= a_{11}r_1 + a_{12}r_2 + \dots + a_{1m}r_m + c_1; \\ r_2 &:= a_{21}r_1 + a_{22}r_2 + \dots + a_{2m}r_m + c_2; \end{aligned}$$

$$r_m := a_{m1}r_1 + a_{m2}r_2 + \dots + a_{mm}r_m + c_m;$$

або $r := Ar + c$, де

$$|A| = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mm} \end{vmatrix},$$

$r = (r_1, r_2, \dots, r_m)$ — вектор-стовпчик, а $c = (c_1, c_2, \dots, c_m)$ — вектор вільних членів.

Елементи множини $\text{ef}(\emptyset, y)$ зводяться в цьому випадку до залежностей типу

$$\gamma_1 r'_1 + \gamma_2 r'_2 + \dots + \gamma_m r'_m = 0,$$

де $r'_i = r_i - c_i$, $i = 1, 2, \dots, m$. Ці елементи еквівалентні, очевидно, рівностям

$$\gamma_1 r_1 + \gamma_2 r_2 + \dots + \gamma_m r_m + \gamma_{m+1} = 0.$$

Ясно, що множина $\text{ef}(\emptyset, y)$ непуста, якщо рівняння $\gamma A = 0$ ($\gamma = (\gamma_1, \gamma_2, \dots, \gamma_m)$) має ненульовий розв'язок, тобто коли система рівнянь

$$\left\{ \begin{array}{l} \gamma_1 a_{11} + \gamma_2 a_{21} + \dots + \gamma_m a_{m1} = 0, \\ \gamma_1 a_{12} + \gamma_2 a_{22} + \dots + \gamma_m a_{m2} = 0, \\ \dots \\ \gamma_1 a_{1m} + \gamma_2 a_{2m} + \dots + \gamma_m a_{mm} = 0 \end{array} \right.$$

має відмінний від нуля розв'язок. Цей розв'язок існує тоді і тільки тоді, коли визначник матриці A дорівнює нулю, тобто коли рядки (стовпчики) матриці A лінійно залежні.

З лінійної алгебри відомо, що множина всіх розв'язків системи $\gamma A = 0$ буде векторним простором, а її фундаментальна система розв'язків — базисом цього простору. Зокрема, якщо всі праві частини виразів $r' := t(r)$ дорівнюють нулю, тобто $t(r) = 0$, то умовимося вважати, що фундаментальна система розв'язків складається з “одиничних” векторів, що мають вигляд $\gamma_1 = (1, 0, \dots, 0)$, $\gamma_2 = (0, 1, \dots, 0)$, ..., $\gamma_m = (0, 0, \dots, 1)$.

Нехай P — множина рівностей, які відповідають фундаментальній системі розв'язків рівняння $\gamma A = 0$. Тоді внаслідок очевидної алгебраїчної замкнутості векторних просторів справедливе таке твердження.

Лема 12.4.1. $C(P) = \text{ef}(\emptyset, y)$, $|P| = |R|$, тобто $C(P)$ — нормальні конгруентності.

Доведення. Нехай P_1, P_2 — базиси і $|P_1| = p, |P_2| = q$. Вияснимо, яка множина буде базисом для $T = C(P_1) \cap C(P_2)$. Покладемо для означеності, що $p \leq q$. Оскільки T — підпростір як $C(P_1)$, так і $C(P_2)$, то довжина T не перевищує величини $\min(p, q) = p$. Нехай $|P_1|, |P_2|$ — матриці, рядки яких складаються з координат векторів, — базисів P_1 і P_2 відповідно. Виконаємо спочатку деякі перетворення, що спрощують будову базисів P_1 і P_2 та пов'язаних з ними матриць.

Виконуючи перетворення, які полягають у додаванні до одного рядка деякої лінійної комбінації других рядків і перестановок стовпчиків, зведемо матрицю $|P_1|$ до такого вигляду:

$$|P'_1| = \begin{vmatrix} 1 & 0 & 0 & \dots & 0 & d'_{1p+1} & \dots & d'_{1m+1} \\ 0 & 1 & 0 & \dots & 0 & d'_{2p+1} & \dots & d'_{2m+1} \\ \dots & \dots \\ \dots & \dots \\ 0 & 0 & 0 & \dots & 1 & d'_{pp+1} & \dots & d'_{pm+1} \end{vmatrix}.$$

Нехай T_1, T_2, \dots, T_k — перетворення, які виконувались над стовпчиками матриці $|P_1|$. Застосуємо T_1, T_2, \dots, T_k до стовпчиків матриці $|P_2|$, тобто розмістимо стовпчики $|P_2|$ так, як вони розміщені в матриці $|P_1|$. Тепер, використовуючи лише перетворення рядків (аналогічні тим, що виконувались над рядками $|P_1|$), зведемо $|P_2|$ до вигляду

$$|P'_2| = \begin{vmatrix} d_{11} & d_{12} & \dots & d_{1q} & \dots & d_{1m+1} \\ 0 & d_{22} & \dots & d_{2q} & \dots & d_{2m+1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & d_{qq} & \dots & d_{qm+1} \end{vmatrix}.$$

Цим процес перетворень базисів закінчується.

Побудова базису PT множини T ґрунтується на такому простому зауваженні.

Нехай $f = (f_1, \dots, f_m, f_{m+1})$ в базисі P_1 і $f' = (f'_1, \dots, f'_{m'}, f_{m+1}')$ в базисі P_2 . Тоді $f \in T$ і $f_i = f'_i, i = 1, 2, \dots, m + 1$, тобто $(f_1, \dots, f_m, f_{m+1}) - (f'_1, \dots, f'_{m'}, f_{m+1}') = 0$. Значить, для побудови множини PT потрібно знайти лінійні комбінації векторів із P_1 і P_2 , які рівні між собою. Пошук таких комбінацій можна виконати в два етапи. На першому етапі, віднімаючи лінійні комбінації векторів із P_1 від векторів із P_2 , добиваємося того, щоб в одержаних векторах перші p координат були нульовими. На другому етапі знаходимо серед одержаних векторів лінійно залежні. Ці залежні век-

тори і дають лінійні комбінації векторів із P_1 і P_2 , які рівні між собою. Вибираючи з кожного такого вектора ті комбінації векторів, які належать одному і тому ж $C(P_i)$, $i = 1, 2$, побудуємо базис множини T .

Оскільки ці побудови досить прості, для більшої переконливості дамо їх формальний виклад.

Перший етап. Віднімаючи i -й рядок матриці $|P_1'|$ від першого, другого, і т.д. i -го рядка матриці $|P_2'|$, помножений на відповідні числа, зведемо $|P_2'|$ до вигляду

$$|P_2''| = \begin{vmatrix} 0 & 0 & \dots & 0 & d'_{1p+1} & \dots & \dots & \dots & d'_{1m+1} \\ 0 & 0 & \dots & 0 & d'_{2p+1} & \dots & \dots & \dots & d'_{2m+1} \\ \dots & \dots \\ \dots & \dots \\ 0 & 0 & \dots & 0 & d'_{pp+1} & \dots & \dots & \dots & d'_{pm+1} \\ \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & d'_{qq} & \dots & d'_{qm+1} \end{vmatrix}.$$

Нехай e_i, e'_i, c_k — вектори, координати яких — рядки матриць $|P_1'|$, $|P_2'|$ і $|P_2''|$ відповідно, а B — матриця, яка одержується із матрицею $|P_2''|$, має вигляд

$$|B| = \begin{vmatrix} d'_{1p+1} & \dots & \dots & \dots & \dots & \dots & d'_{1m+1} \\ d'_{2p+1} & \dots & \dots & \dots & \dots & \dots & d'_{2m+1} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ d'_{pp+1} & \dots & \dots & \dots & \dots & \dots & d'_{pm+1} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & d'_{qq} & \dots & d'_{qm+1} \end{vmatrix}.$$

При таких позначеннях виконуються рівності:

$$\begin{aligned} c_1 &= e'_1 - d_{11}e_1 - \dots - d_{1p}e_p, \\ c_2 &= e'_2 - d_{22}e_2 - \dots - d_{2p}e_p, \\ &\dots \\ c_p &= e'_p - d_{pp}e_p, \\ c_{p+1} &= e'_{p+1}, \\ &\dots \\ c_q &= e'_q. \end{aligned} \tag{12.4.1}$$

Другий етап. Припустимо, що перші $t - 1$ векторів c_i , $t < q$, $i = 1, 2, \dots, q$, лінійно незалежні. Тоді

$$c_j = d_{j1}c_1 + \dots + d_{jt-1}c_{t-1} \tag{12.4.2}$$

для всіх $j = t, \dots, q$, або $c_j - d_{j1}c_1 - \dots - d_{jt-1}c_{t-1} = 0$. Підставимо замість $c_i, c_j, i = 1, 2, \dots, t-1, j = 1, 2, \dots, q$ їх вирази із (12.4.1), тоді (12.4.2) зводиться до такого вигляду:

$$(e'_j - d_{j1}e'_1 - \dots - d_{jt-1}e'_{t-1}) + (h_1e_1 + \dots + h_pe_p) = 0.$$

Нехай $d_j = (e'_j - d_{j1}e'_1 - \dots - d_{jt-1}e'_{t-1})$ — перший доданок у цій рівності, а PT — множина, яка складається з $d_j, j = 1, 2, \dots, q$.

Основну властивість множини PT дає таке твердження.

Лема 12.4.2. $C(PT) = C(P_1) \cap C(P_2)$.

Доведення. Очевидно, що $PT \subseteq C(P_1) \cap C(P_2)$ і, значить, $C(PT) \subseteq C(P_1) \cap C(P_2)$. Нехай d — довільний елемент із $T = C(P_1) \cap C(P_2)$. Тоді $d = a_1e_1 + \dots + a_pe_p$ і $d = b_1e'_1 + \dots + b_qe'_q$. Оскільки $a_1e_1 + \dots + a_pe_p = b_1e'_1 + \dots + b_qe'_q$, то $a_1 = b_1d_{11}, a_2 = b_1d_{12} + b_2d_{22}, \dots, a_p = b_1d_{1p} + \dots + b_pd_{pp}$. Використовуючи ці рівності, можна записати, що $b_1e'_1 + \dots + b_qe'_q - a_1e_1 - \dots - a_pe_p = b_1c_1 + \dots + b_qc_q$. Покажемо, що d — лінійна комбінація векторів $d_j, j = t, \dots, q$. Із (12.4.2) випливає, що вектор $b_1c_1 + \dots + b_qc_q = (b_1 + b_td_{tt} + \dots + b_qd_{qq})c_1 + \dots + (b_{t-1} + b_td_{tt-1} + \dots + b_qd_{qt-1})c_{t-1} = 0$. Оскільки c_1, \dots, c_{t-1} лінійно незалежні, то $(b_t + b_td_{tt} + \dots + b_qd_{qt}) = 0$. Значить, $d = b_td_{tt} + \dots + b_qd_{qq}$ і $d \in C(PT)$. Оскільки елемент d довільний, то $C(PT) \supseteq C(P_1) \cap C(P_2)$, звідки одержуємо рівність $C(PT) = C(P_1) \cap C(P_2)$.

Лема доведена.

Для побудови векторів d_j із PT зведемо матрицю B до діагональної форми. Позначимо одержану матрицю B' , і нехай i_1, \dots, i_k — номери нульових рядків в B' , а C_1, C_2, \dots, C_N — перетворення, які застосувались до B в процесі побудови B' . Застосуємо ці перетворення до $|P_2'|$. Тоді вектори d_j , координати яких складають рядки $i_j, j = 1, 2, \dots, k$, становитимуть собою базис множини T .

Наслідок 12.4.1. Якщо $C(PT) = C(P_1) \cap C(P_2) \cap \dots \cap C(P_k)$, то побудова PT не залежить від порядку вибору $P_i, i = 1, 2, \dots, k$.

Наслідок 12.4.2. $C(PT)$ — нормальна конгруентність.

Доведення. Нехай $|PT|$ має діагональний вигляд і $(\alpha_{ii}, \dots, \alpha_{ii}, \dots, \alpha_{im+1})$ — вектор-рядок $|PT|$, в якому $\alpha_{ii} = 1, \alpha_{il} = \alpha_{in} = \dots = \alpha_{ii-1} = 0$. Побудуємо оператор присвоювання $y = (r_1 := t_1(r), \dots, r_m := t_m(r))$, в якому $t_i(r) = -\alpha_{il}r_1 - \dots - \alpha_{ii-1}r_{i-1} - \alpha_{ii+1}r_{i+1} - \dots - \alpha_{im}r_m - \alpha_{im+1}$, якщо $d_i = (\alpha_{il}, \dots, \alpha_{ii}, \dots, \alpha_{im+1}) \in PT$, і $t_i(r) = r_i$, в протилежному випадку. Очевидно, що $C(PT) = \text{ef}(\emptyset, y)$.

Із наслідку 12.4.2 і теорем 12.2.3, 12.2.4, 12.2.7 випливає справедливість такого твердження.

Наслідок 12.4.3. Алгоритм МВА (МНА) завершує свою роботу за скінченне число кроків. Часова складність алгоритму МВА (МНА) пропорційна величині $n * m^{3,46}$.

Д о в е д е н я. Число змін множини співвідношень у кожному стані обмежене величиною m . Тоді загальна кількість ітерацій алгоритму становить $n * m$.

На кожній ітерації алгоритму розв'язується система з m лінійних рівнянь і будеється, якщо потрібно, базис перетину. Обидві операції вимагають часу $O(m^{2,81})$ (а враховуючи останні досягнення теорії складності алгоритмів, цю оцінку можна понизити до $O(m^{2,46})$) [4].

Таким чином, остаточна оцінка часової складності виражається величиною $O(n * m^{3,46})$, що й потрібно було довести.

2. ВІЛЬНІ АБЕЛЕВІ ГРУПИ І КОМУТАТИВНІ НАПІВГРУПИ

Нехай $T_D(R)$ — вільна абелева група. Тоді її можна розглядати як скінченно-породжений модуль над кільцем цілих чисел \mathbf{Z} . У цьому випадку елементи $\text{ef}(\emptyset, y)$ мають вигляд

$$\gamma_1 r'_1 + \gamma_2 r'_2 + \dots + \gamma_m r'_m = 0, \quad (12.4.3)$$

де $\gamma_i \in \mathbf{Z}$, $i = 1, 2, \dots, m$. Можливість зведення задачі пошуку інваріантів для програм, що розглядаються над вільними абелевими групами, до такої самої задачі у випадку, коли $T_D(R)$ — векторний простір, дає така теорема.

Теорема 12.4.1. Всяка область цілісності ізоморфно вкладається в поле своїх дробів [42].

Д о в е д е н я. Кільце цілих чисел \mathbf{Z} є областю цілісності, а полем його дробів є поле раціональних чисел. Значить, кільце \mathbf{Z} можна ізоморфно вкладти в поле раціональних чисел і виконувати пошук інваріантів, вважаючи, що $T_D(R)$ — векторний простір над полем раціональних чисел. Побудувавши інваріанти, їх необхідно привести до вигляду (12.4.3). Для цього потрібно виконати перетворення, які знищують знаменники в коефіцієнтах, якщо вони в них є.

У випадку, коли $T_D(R)$ — вільна комутативна напівгрупа, елементи $\text{ef}(\emptyset, y)$ мають вигляд

$$\gamma_1 r'_1 + \dots + \gamma_k r'_k = \gamma_{k+1} r'_{k+1} + \dots + \gamma_m r'_m, \quad (12.4.4)$$

де γ_i — цілі невід'ємні числа.

Знову використовуючи засіб ізоморфного вкладення однієї

алгебраїчної структури в іншу, як і раніше, можна звести задачу пошуку інваріантів для програм, що розглядаються над вільними комутативними напівгрупами, до пошуку таких, коли $T_D(R)$ — вільна абелева група. Підставу для цього дає така теорема.

Теорема 12.4.2. *Всяка комутативна напівгрупа, в якій виконується закон скорочення, ізоморфно вкладається в абелеву групу [42].*

Додавання коефіцієнтів в $T_D(R)$ зводиться до додавання коефіцієнтів при однакових змінних $r_i \in R$. Отже, закон скорочення в $T_D(R)$ має місце, і можна будувати інваріанти описаним вище способом. Після цього одержані інваріанти необхідно привести до вигляду (12.4.4). Таким чином, справедлива теорема.

Теорема 12.4.3. *Для всякої U-Y-програми, що розглядається над: а) векторним простором; б) вільною абелевою групою; в) вільною комутативною напівгрупою, алгоритм МВА (МНА) завершує свою роботу після скінченного числа кроків. Часова оцінка складності алгоритму МВА пропорційна величині $n * m^{3,46}$, $n = |A|$, $m = |R|$.*

Зазначимо, що коли умови $u \in U$, які зустрічаються в U-Y-програмі, є рівностями, то за допомогою алгоритму МВА (МНА) можна враховувати такі умови. Дійсно, при побудові базису $\text{ef}(M, u)$ для переходу (a, u, y, b) врахування умови u виконується шляхом додавання до базису множини M умови u з наступною перевіркою належності u до M . Якщо $u \in M$, то базис M не змінюється, інакше базис M розширяється шляхом додавання до нього умови u . Якщо ж перехід включає заперечення деякої умови, тобто має вигляд $(a, \neg u, y, b)$, де u — рівність, то $\text{ef}(M, u)$ будується звичайним способом.

3. ПРИКЛАДИ

Розглянемо приклади деяких U-Y-програм і, застосовуючи до них алгоритм МВА, побудуємо для них множини інваріантних співвідношень.

1. Розглянемо приклад програми множення квадратних матриць порядку n — MULT(X, Y, n).

Елементи початкових матриць X і Y розміщені в двовимірних масивах, а результат розміщується в масиві Z . При множенні матриць виконується перехід від двовимірних масивів до одновимірних. Наведена нижче U-Y-програма (рис. 12.4.1), алгебра даних якої вважається абсолютно вільною, записана в більш загальній мові, ніж мова стандартних U-Y-програм, оскільки вона містить оператори оброблення масивів.

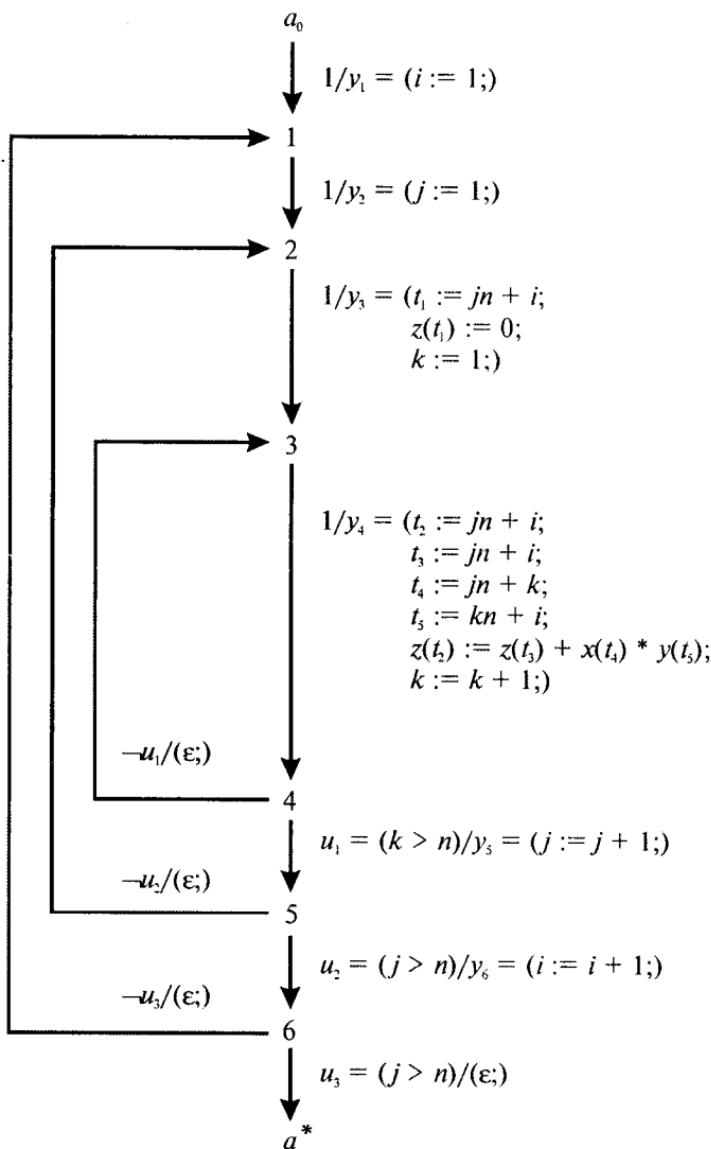


Рис. 12.4.1. U - Y -програма $MULT(X, Y, n)$

Але від такої U - Y -програми можна перейти до звичайної стандартної U - Y -програми, якщо присвоювання елементам масиву і умови перевірки, пов'язані з масивами, ігнорувати (рис. 12.4.2).

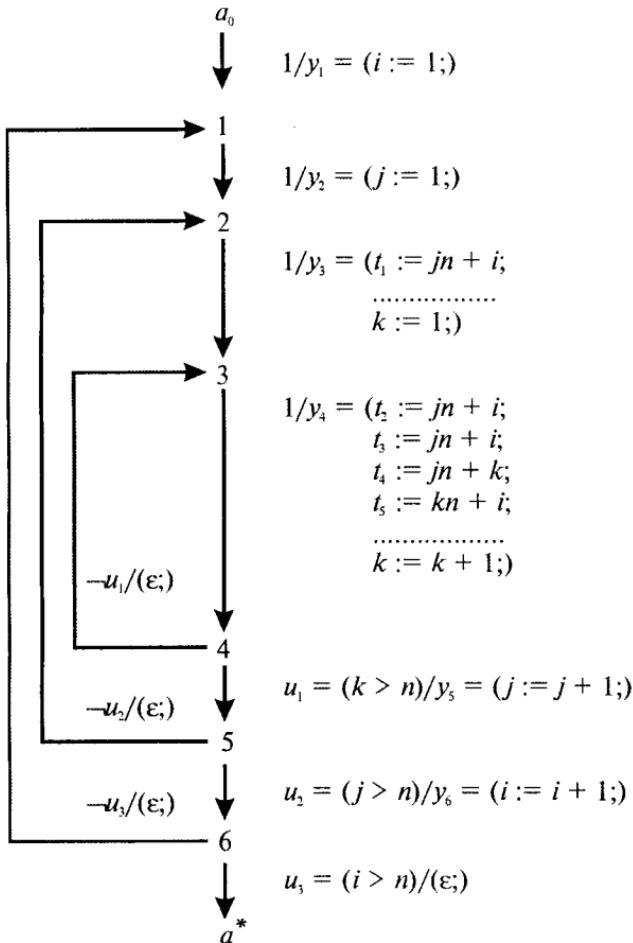


Рис. 12.4.2. Стандартна U - Y -програма $MULT(X, Y, n)$

У цій програмі маємо:

- $R = \{i, j, k, t_1, t_2, t_3, t_4, t_5\};$
- $A = \{a_0, 1, 2, 3, 4, 5, 6, a^*\};$
- $Y = \{y_1, y_2, y_3, y_4, y_5, y_6, \varepsilon\};$
- $U = \{1, u_1, u_2, u_3, -u_1, -u_2, -u_3\};$
- $S = \{(a_0, 1, y_1, 1), (1, 1, y_2, 2), (2, 1, y_3, 3), (3, 1, y_4, 4),$
 $(4, -u_1, \varepsilon, 3), (4, u_1, y_5, 5), (5, -u_2, \varepsilon, 2), (5, u_2, y_6, 6),$
 $(6, -u_3, \varepsilon, 1), (6, u_3, \varepsilon, a^*)\};$
- $N_{a_0} = \emptyset.$

На першому етапі роботи алгоритму МВА виконується така послідовність дій.

- $N_{a_0} = \emptyset; C = \{a_0\}.$

2. $\alpha(1) = \dots = \alpha(a_0) = 0$.

3. $\alpha(a^*) = 1$.

4. Будуємо базис $\text{ef}(N_{a_0}, y_1)$, він, очевидно, складається із
єдиного співвідношення $i = 1$, тобто $N_1 = \{i = 1\}$.

5. $\alpha(1) = 1; C = \{1\}$.

6. Будуємо базис $\text{ef}(N_1, y_2)$, очевидно, що $N_2 = \{i = 1, j = 1\}$.

7. $\alpha(2) = 1; C = \{2\}$.

8. Будуємо базис $\text{ef}(N_2, y_3)$, знову очевидно, що $N_3 = \{i = 1, j = i, k = i, t_1 = in + i\}$.

9. $\alpha(3) = 1; C = \{3\}$.

10. Будуємо базис $\text{ef}(N_3, y_4)$:

$i := 1;$	$i := 1;$
$j := 1;$	$j := 1;$
$k := 1;$	$k := 2;$
$t_1 := in + i;$	$t_1 := 1n + 1;$
$t_2 := jn + i;$	$t_2 := 1n + 1;$
$t_3 := jn + i;$	$t_3 := 1n + 1;$
$t_4 := jn + k;$	$t_4 := 1n + 1;$
$t_5 := kn + i;$	$t_5 := 1n + 1;$
$k := k + 1;$	

звідси одержуємо базис N_4 :

$N_4 = \{i = 1, j = 1, k = 2, t_1 = in + i, t_2 = t_1, t_3 = t_1, t_4 = t_1, t_5 = t_1\}$.

11. $\alpha(4) = 1, C = \{4\}$.

12. Будуємо базис $\text{ef}(N_4, y_5)$:

$i := 1;$	$i := 1;$
$j := 1;$	$j := 2;$
$k := 2;$	$k := 2;$
$t_1 := in + i;$	$t_1 := 1n + 1;$
$t_2 := t_1;$	$t_2 := 1n + 1;$
$t_3 := t_1;$	$t_3 := 1n + 1;$
$t_4 := t_1;$	$t_4 := 1n + 1;$
$t_5 := t_1;$	$t_5 := 1n + 1;$
$j := j + 1;$	

звідси одержуємо базис N_5 :

$N_5 = \{i = 1, j = 2, k = j, t_1 = in + i, t_2 = t_1, t_3 = t_1, t_4 = t_1, t_5 = t_1\}$.

13. $\alpha(5) = 1, C = \{5\}$.

14. Будуємо базис $\text{ef}(N_5, y_6)$, не вдаючись до подробиць (оскільки вони майже аналогічні тим, що були в п.12)), знаходимо

$N_6 = \{i = 2, j = i, k = j, t_1 = 1n + 1, t_2 = t_1, t_3 = t_1, t_4 = t_1, t_5 = t_1\}$.

15. $\alpha(6) = 1, C = \{6\}$.

16. Базис N_a не будуємо, оскільки $\alpha(a^*) = 1$ (див. п.3).

На цьому перший етап роботи алгоритму завершується.

Другий етап починається при $C = \{1, 2, 3, 4, 5, 6, a^*\}$.

1. Вибираємо стан 1 із C , тобто $C = \{2, 3, 4, 5, 6, a^*\}$.

Послідовно маємо $N = N_1$, $\text{ef}(N_6, \epsilon) = N_6$. Будуємо за базисами N і N_6 нове значення базису N . Але це значення, очевидно, дорівнює пустій множині, оскільки єдине співвідношення $i = 1$ не є наслідком співвідношень із N_6 . Оскільки $N \neq N_1$, то N_1 покладаємо рівним N , тобто пустій множині.

2. $N_1 = \emptyset$, $C = \{2, 3, 4, 5, 6, a^*\}$;

3. Вибираємо стан 2 із C , тобто $C = \{3, 4, 5, 6, a^*\}$, $N = N_2$ і будуємо базиси $\text{ef}(N_5, \epsilon)$ і $\text{ef}(N_1, y_2)$. Перший дорівнює N_5 , а $\text{ef}(N_1, y_2) = \{j = 1\}$. Нове значення змінної N дорівнює, з тих же причин, що і в попередньому випадку, пустій множині. Оскільки $N \neq N_2$, то N_2 стає пустою множиною, а $C = \{3, 4, 5, 6, a^*\}$.

4. Вибираємо стан 3 із C , тобто $C = \{4, 5, 6, a^*\}$, $N = N_3 = \{i = 1, j = i, k = i, t_1 = in + i\}$. Будуємо базиси $\text{ef}(N_2, y_3)$ і $\text{ef}(N_4, \epsilon)$. Перший дорівнює $\text{ef}(\emptyset, y_3) = \{k = 1; t_1 = jn + i\}$, а другий — $N_4 = \{i = 1, k = 2, t_1 = jn + i, t_2 = t_1, t_3 = t_1, t_4 = t_1, t_5 = t_1\}$.

За базисами N_4 і $\text{ef}(\emptyset, y_3)$ будуємо базис перетину

$$N_4 = C(k = 1, t_1 = jn + i) \cap C(i = 1, j = i, k = 2, t_1 = jn + i).$$

Звідси одержуємо, що шуканий базис складається з єдиного співвідношення $t_1 = jn + i$, тобто $N = \{t_1 = jn + i\}$. Оскільки $N \neq N_3$, то N_3 покладаємо рівним N , а $C = \{4, 5, 6, a^*\}$.

5. Вибираємо стан 4 із C , тобто $C = \{5, 6, a^*\}$, і будуємо базис $\text{ef}(N_3, y_4)$:

$$\begin{array}{lll} t_1 := jn + i; & & k := k + 1; \\ t_2 := jn + i; & \text{після суперпозиції} & t_1 := jn + i; \\ t_3 := jn + i; & \text{i упорядкування} & t_2 := jn + i; \\ t_4 := jn + k; & & t_3 := jn + i; \\ t_5 := kn + i; & & t_4 := jn + k; \\ k := k + 1; & & t_5 := kn + i, \end{array}$$

Звідки одержуємо $N_4 = \{t_1 = jn + i, t_2 = t_1, t_3 = t_1\}$.

Будуємо базис перетину N за множинами $C(\{i = 1, j = 1, k = 2, t_1 = jn + i, t_2 = t_1, t_3 = t_1, t_4 = t_1, t_5 = t_1\})$ і $C(N_4)$. В результаті одержуємо базис $N = \{t_1 = jn + i, t_2 = t_1, t_3 = t_1\}$. Оскільки базис $N \neq N_4$, то N_4 стає рівним N , а $C = \{3, 5, 6, a^*\}$.

6. Вибираючи 3 із C і повторюючи побудову базисів, одержуємо $N = N_3 = \{t_1 = jn + i\}$, а $C = \{5, 6, a^*\}$.

7. Вибираємо 5 із C і N надаємо значення N_5 . Будуємо базис $\text{ef}(N_4, y_5)$:

$$\begin{array}{lll} t_1 := jn + i; & & j := j + 1; \\ t_2 := t_1; & \text{після суперпозиції} & t_1 := jn + i; \end{array}$$

$$\begin{array}{lll} t_3 := t_1; & \text{i упорядкування} & t_2 := jn + i; \\ j := j + 1; & & t_3 := jn + i. \end{array}$$

Звідси одержуємо $N_5 = \{t_1 = jn + i, t_2 = t_1, t_3 = t_1\}$. Оскільки $N \neq N_5$, то $C = \{2, 6, a^*\}$ і будуємо базис перетину за базисами N і N_5 , тобто $N_5 = \{t_2 = t_1, t_3 = t_1\}$.

8. Вибираємо стан 2 із C , тобто $C = \{6, a^*\}$. Оскільки $N_2 = \emptyset$, то $N = N_2$ і $C = \{6, a^*\}$.

9. Вибираємо стан 6 із C , N стає рівним N_6 і будуємо $\text{ef}(N_6, y_6)$:

$$\begin{array}{lll} t_2 := t_1; & t_2 := t_1; \\ t_3 := t_1; & \text{після суперпозиції} & t_3 := t_1; \\ i := i + 1; & \text{nічого не змінюється} & i := i + 1; \end{array}$$

звідси одержуємо $N = C(N) \cap C(N_6) = \{t_2 = t_1, t_3 = t_1\}$. Оскільки $N \neq N_6$, то N_6 стає рівним N , а $C = \{1, a^*\}$.

10. Вибираємо стан 1 із C , тобто $C = \{a^*\}$. Оскільки $N_1 = \emptyset$, то нічого не змінюється.

11. Вибираємо a^* із C , тобто $C = \emptyset$, і базис $N_{a^*} = N_6 = \{t_2 = t_1, t_3 = t_1\}$. Але оскільки $C = \emptyset$, то алгоритм завершує свою роботу.

2. Розглянемо вже знайомий нам з розділу 2 (§ 2.4) приклад U - Y -програми $\text{УМН}(x, y)$ — програми множення цілих додатних чисел x і y . Для зручності зобразимо цю програму у вигляді графа (рис. 12.4.3) і простежимо роботу алгоритму МВА для цієї програми. Алгебра даних вважається вільною абелевою напівгрупою.

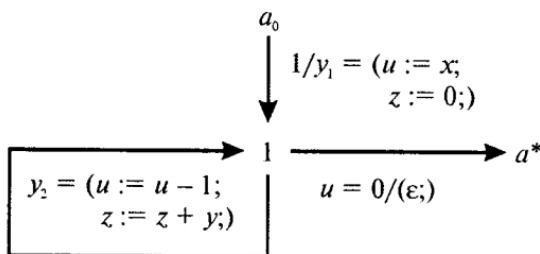


Рис. 12.4.3. U - Y -програма $\text{УМН}(x, y)$

Для цієї програми маємо:

- $R = \{u, z\}$;
- $A = \{a_0, 1, a^*\}$;
- $Y = \{y_1, y_2, ε\}$;
- $U = \{1, u_1 = (u = 0), -u_1\}$;
- $S = \{(a_0, 1, y_1, 1), (1, -u_1, y_2, 1), (1, u_1, ε, a^*)\}$;
- $N_0 = \emptyset$.

U-Y-програма УМН(x, y)

Перший етап роботи алгоритму МВА:

1. $N_{a_0} = \emptyset$.
2. $\alpha(1) = 0; \alpha(a^*) = 0$.
3. $\alpha(a^*) = 1$.
4. Будуємо базис $\text{ef}(N_{a_0}, y_1)$:

$$u := x; \text{ суперпозиція } u' = u - x = 0; \\ z := 0; \text{ нічого не змінює } z' = z = 0.$$

Значить, фундаментальна система розв'язків має вигляд: $(1, 0), (0, 1)$, тобто базис $\text{ef}(N_{a_0}, y_1)$ складає множина $N_1 = \{u = x, z = 0\}$, матриця якого

$$P(1, 1) = \begin{vmatrix} 1 & 0 & -x \\ 0 & 1 & 0 \end{vmatrix}.$$

Після цього $\alpha(1)$ дорівнюватиме 1, а $C = \emptyset$. Таким чином, перший етап завершується.

Другий етап роботи алгоритму МВА починається при $C = \{1, a^*\}$ і $N = N_1$.

1. Вибираємо стан 1 і будуємо базис множини $\text{ef}(N_1, y_2)$:

$$u := x; \\ z := 0; \text{ після } u := x - 1; \quad u' = u - x + 1 = 0; \\ u := u - 1; \text{ суперпозиції } z := y; \text{ або } z' = z - y = 0. \\ z := z + y;$$

Значить, базис N_1 складають співвідношення $u = x - 1, z = y$, а його матриця має вигляд

$$P(1, 2) = \begin{vmatrix} 1 & 0 & -x + 1 \\ 0 & 1 & -y \end{vmatrix}.$$

2. Оскільки $N \neq N_1$, то будуємо базис перетину:

$$\left. \begin{array}{l} P(1, 1) = \begin{vmatrix} 1 & 0 & -x \\ 0 & 1 & 0 \end{vmatrix} \\ P(1, 2) = \begin{vmatrix} 1 & 0 & -x + 1 \\ 0 & 1 & -y \end{vmatrix} \end{array} \right\} \Rightarrow \begin{vmatrix} 0 & 0 & 1 \\ 0 & 0 & -y \end{vmatrix} \Rightarrow e_2 + ye_1.$$

Отже, одержуємо таке співвідношення: $z - y + (u - x + 1)y = z + uy - xy = 0$ або $z + uy = xy$. Множина C матиме вигляд $\{1, a^*\}$.

3. Вибираємо стан 1 і будуємо базис N_1 для $\text{ef}(N_1, y_2)$ при $N_1 = \{z + uy = xy\}$:

$$z := -uy + xy; \quad u := u - 1; \\ u := u - 1; \quad \text{після суперпозиції } z := -uy + xy + y; \text{ або}$$

$$z := z + y;$$

$$u' = u + 1 = u;$$

$$z' = z - xy - y = -uy,$$

звідки одержуємо систему

$$\gamma_1 u' + \gamma_2 z' = 0,$$

рівносильну системі

$$\gamma_1 u - \gamma_2 uy = 0,$$

розв'язуючи яку, одержуємо $(y, 1)$. Цьому розв'язку відповідає співвідношення $z - xy - y + y(u + 1) = z + uy - xy = 0$ або $z + uy = xy$.

4. Оскільки базиси збігаються, то $C = \{a^*\}$, вибираємо a із C і будуємо $\text{ef}(N_1, e)$. Побудова базису цієї множини зводиться до додавлення співвідношення $u = 0$ до N_1 , оскільки воно не є наслідком співвідношень із N_1 . Таким чином, в стані a^* мають місце такі співвідношення:

$$u = 0, z + uy = xy.$$

Наслідком цієї множини співвідношень є, зокрема, співвідношення $z = xy$, на основі якого можна зробити висновок, що програма УМН(x, y) правильно обчислює результат.

Задачі і вправи

1. Запишіть U - Y -програму ДІЛ(z_1, z_2) у вигляді регулярного виразу і складеного об'єкта.

2. Упевнітесь, що на другому етапі роботи алгоритму МВА для U - Y -програми ДІЛ(z_1, z_2) множини співвідношень в станах не залежать від вибору станів із множини C при їх генерації.

3. Застосуйте алгоритм МВА до наведеної нижче U - Y -програми і побудуйте для неї інваріантні співвідношення:

ДІЛ1(x, y)

початок.

$$q := 0;$$

$$r := x;$$

поки $r > y$ **виконувати**

$$r := r - y;$$

$$q := q + 1;$$

кц;

стоп;

кінець.

4. Побудуйте базис множини співвідношень $\text{ef}(N_3, y_4)$ для програми множення матриць і простежте всі кроки алгоритму генерації співвідношень для даного переходу.

5. Використовуючи інформацію про вершини, які складають класи A_1, A_2, \dots, A_m (A'_1, A'_2, \dots, A'_m), наведіть варіант алгоритму, ефективнішого за алгоритм **ПЕРЕТИН**.

6. Побудуйте множину інваріантів для U - Y -програми ділення цілих чисел з остачею — програма $\text{ДІЛ}(z_1, z_2)$ (алгебра даних — вільна абелева група).

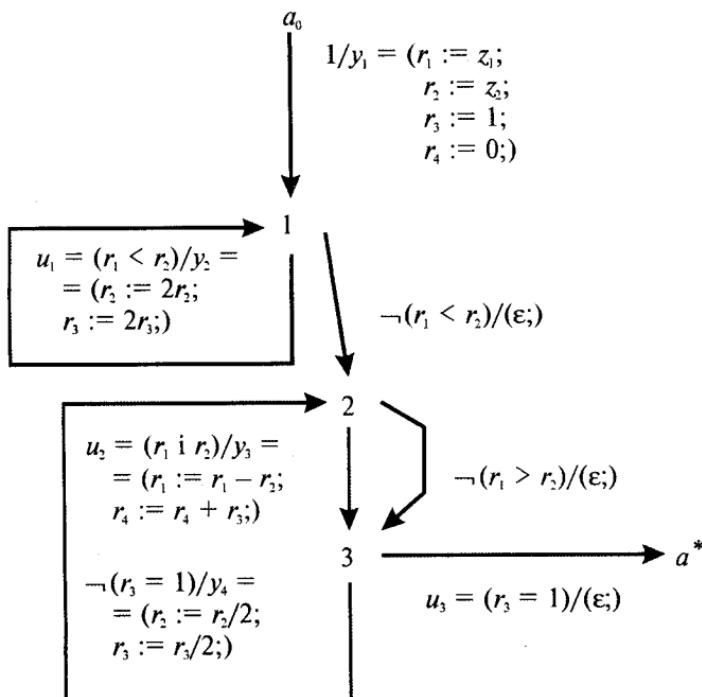


Рис.12.4.4. Граф переходів U - Y -програми $\text{ДІЛ}(z_1, z_2)$

СПИСОК ЛИТЕРАТУРИ

1. Александров П.С. Введение в теорию групп. — М.: Наука, 1980. — 144 с.
2. Александров А.Д., Нецветаев Н.Ю. Геометрия. — М.:Наука, 1990. — 671 с.
3. Анисимов А.В. О групповых языках // Кибернетика. — 1972. — № 4. — С. 18—24.
4. Axo A., Хопкрофт Дж., Ульман Дж. Анализ и построение вычислительных алгоритмов. — М.: Мир, 1975. — 457 с.
5. Биркгоф Г., Барти Т. Современная прикладная алгебра. — М.: Мир, 1976. — 400 с.
6. Компьютерная алгебра (Символьные и алгебраические вычисления) // Под ред. Б. Бухбергера, Дж. Коллинза, Р. Лооса. — М.: Мир, 1986. — 386 с.
7. Van der Варден Б.Л. Алгебра. — М.: Наука, 1976. — 648 с.
8. Вирт Н. Систематическое программирование. Введение. — М.: Мир, 1977. — 183 с.
9. Васильев Ю.Л., Ветухновский Ф.Я., Глаголев В.В. и др. Дискретная математика и математические вопросы кибернетики. — М.: Наука, 1974. — Т.1 — 311 с.
10. Виленкин Н.Я. Комбинаторика. — М.: Наука, 1969. — 161 с.
11. Виленкин Н.Я. Популярная комбинаторика. — М.: Наука, 1975. — 207 с.
12. Вирт Н. Систематическое программирование. — М: Мир, 1978. — 183 с.
13. Гизбург С. Математическая теория контекстно-свободных языков. — М.: Мир, 1970. — 326 с.
14. Гладкий А.В. Формальные грамматики и языки. — М.: Наука, 1973.
15. Глушков В.М. Синтез цифровых автоматов. — М.: Физматгиз, 1962. — 562 с.
16. Глушков В.М. Введение в кибернетику. — Киев: Изд-во АН УССР. — 1964. — 324 с.
17. Глушков В.М., Летичевский А.А. Теория дискретных преобразователей // Избр. вопр. алгебры и логики. — Новосибирск: Наука, 1973. — С. 5—39.
18. Глушков В.М., Летичевский А.А., Годлевский А.Б. Методы математической биологии. Кн.6. Методы синтеза математических моделей

биологических систем: Уч. пособие для вузов. — Киев: Вища шк., 1983. — 264 с.

19. Глушкин В.М., Цейтлин Г.Е., Ющенко Е.Л. Алгебра, языки, программирование. — Киев: Наук. думка, 1985. — 431 с.

20. Годлевский А.Б., Кривой С.Л. О проектировании эффективных алгоритмов приведения автоматов для некоторых отношений эквивалентности // Кибернетика. — 1989. — № 6. — С. 37–43.

21. Годлевский А.Б., Кривой С.Л. Трансформационный синтез эффективных алгоритмов с учетом дополнительных спецификаций // Кибернетика. — 1986. — № 6. — С. 34–43.

22. Годлевский А.Б., Капитонова Ю.В., Кривой С.Л., Летичевский А.А. Итеративные методы анализа программ // Кибернетика. — 1989. — № 2. — С. 9–19.

23. Гринченко Т.О., Стогний А.О. Машинный интеллект и новые информационные технологии. — Киев: Манускрипт, 1993. — 164 с.

24. Грэй Р. Логика, алгебра и базы данных. — М.: Машиностроение, 1989. — 359 с.

25. Гудстейн Р.Л. Рекурсивный математический анализ. — М.: Наука, 1970. — 472 с.

26. Дэвэнпорт Сире, Турнье. Компьютерная алгебра. — М.: Мир, 1990.

27. Евстигнеев В.А., Касьянов В.Н. Теория графов (Алгоритмы обработки деревьев). — Новосибирск: Наука, 1994. — 360 с.

28. Ежов И.И., Скороход А.И., Ядренко М.И. Элементы комбинаторики. — М.: Наука, 1975. — 79 с.

29. Емеличев В.А., Мельников О.И. и др. Лекции по теории графов. — М.: Наука, 1990. — 382 с.

30. Калужинин Л.А. Введение в общую алгебру. — М.: Наука, 1973. — 447 с.

31. Клини С. Введение в метаматематику. — М.: ИЛ, 1957. — 674 с.

32. Клини С. Математическая логика. — М.: Мир, 1973. — 480 с.

33. Кон П. Универсальная алгебра. — М.: Мир, 1968. — 351 с.

34. Котов В.Е., Сабельфельд В.К. Теория схем программ. — М.: Наука, 1991. — 247 с.

35. Кривой С.Л. Об алгоритме построения базиса пересечения конечно-порожденных свободных групп // Кибернетика. — 1982. — № 4. — С. 5–10.

36. Кривой С.Л. Об алгоритмах построения конгруэнтных замыканий конечных автоматов // Кибернетика и систем. анализ. — 1994. — № 1. — С. 34–44.

37. Кривой С.Л. Проблема унификации в эквациональных теориях // Кибернетика и систем. анализ. — 1997. — № 6. — С. 143–172.

38. Кук Д., Бейз Г. Компьютерная математика. — М.: Мир, 1990. — 383 с.

39. Куратовский К., Мостовский А. Теория множеств. — М.: Мир, 1970. — 416 с.

40. Курош А.Г. Теория групп. — М.: Наука, 1967. — 648 с.

41. Курош А.Г. Общая алгебра. Лекции 1969–70 гг. — М.: Наука, 1972. — 159 с.

42. Курош А.Г. Лекции по общей алгебре. — М.: Наука, 1972. — 399с.
43. Лавров И.А., Максимова Л.Л. Задачи по теории множеств, математической логике и теории алгоритмов. — М.: Наука, 1975. — 239 с.
44. Летичевский А.А., Годлевский А.Б., Кривой С.Л. Об эффективном алгоритме построения базиса подгруппы свободной группы // Кибернетика. — 1981. — № 3. — С. 107–116.
45. Летичевский А.А., Кривой С.Л. О реализации алгоритма Нильсена в системе алгебраического программирования АПС–1 // Кибернетика. — 1994. — № 5. — С. 48–53.
46. Линдон Р. Заметки по логике — М.: Мир, 1968. — 128 с.
47. Лупанов О.Б. О сравнении двух типов источников // Пробл. кибернетики. — М.: Физматгиз. — 1963. — Вып. 2. — С. 321–326.
48. Луцкий Г.М., Кривий С.Л., Печурін М.К. Основи дискретної математики. — Київ: ВІПОЛ, 1995. — 252 с.
49. Магнус В., Каррас А., Солитэр Д. Комбинаторная теория групп. — М.: Мир, 1974. — 456 с.
50. Мальцев А.И. Алгебраические системы. — М.: Наука, 1970. — 370 с.
51. Мальцев А.И. Алгоритмы и рекурсивные функции. — М.: Наука, 1986. — 367 с.
52. Марков А.А., Нагорный Н.М. Теория алгорифмов — М.: ФАЗИСТ, 1996. — 448 с.
53. Мендельсон Э. Введение в математическую логику. — М.: Мир, 1988. — 320 с.
54. Новиков П.С. Элементы математической логики. — М.: Наука, 1973. — 399 с.
55. Озкарахан Э. Машины баз данных и управление базами данных. — М.: Мир, 1989. — 695 с.
56. Оре О. Теория графов. — М.: Наука, 1980. — 336 с.
57. Павленко В.А. Комбинаторная проблема Поста для двух пар слов. — Киев, 1985. — 65 с. — (Препр. / АН УССР. Ин-т математики).
58. Плоткин Б.И. Универсальная алгебра, алгебраическая логика и базы данных. — М.: Наука, 1991. — 446 с.
59. Пост Е. Конечные комбинаторные процессы, формулировка 1 // Машины Поста / Под ред. В.А. Успенского. — М.: Наука, 1979. — С. 89–95.
60. Райзер Г. Комбинаторная математика. — М.: Мир, 1966. — 154 с.
61. Редько В.Н. Некоторые вопросы теории языков // Кибернетика. — 1965. — № 4.
62. Савельев Л.Я. Комбинаторика и вероятность. — Новосибирск: Наука, 1975. — 452 с.
63. Скорняков Л.А. Элементы теории структур. — М.: Наука, 1982. — 158 с.
64. Справочная книга по математической логике. Теория рекурсии. — М.: Мир, 1983. — Т.3.
65. Столл Р. Множества, логика, аксиоматические теории. — М.: Просвещение, 1968. — 230 с.
66. Тейз А., Грибомон П., Луи Ж. и др. Логический подход к искусственному интеллекту. — М.: Мир, 1990. — 429 с.

67. Тейз А., Грибомон П., Юлен А., Пирот А. и др. Логический подход к искусственному интеллекту (От модальной логики к логике баз данных). — М.: Мир, 1998. — 494 с.
68. Уилсон Р. Введение в теорию графов. — М.: Мир, 1977. — 207 с.
69. Феллер В. Введение в теорию вероятностей. — М.: Мир, 1964. — Т.1. — 376 с.
70. Филд А., Харрисон П. Функциональное программирование. — М.: Мир, 1993. — 638 с.
71. Форд Л.Р., Фалкерсон Д.Р. Потоки в сетях — М.: Мир, 1966. — 229 с.
72. Френкель А., Бар-Хиллел И. Основания теории множеств. — М.: Мир, 1966. — 328 с.
73. Холл М. Комбинаторика. — М.: Мир, 1970. — 234 с.
74. Хомский Н. О некоторых формальных свойствах грамматик // Киберн. сборник. — 1962. — № 5. — С. 37–123.
75. Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. — М.: Наука, 1983. — 256 с.
76. Artificial Intelligence and Symbolic Mathematical Computing // Lecture Notes in Comput. Sci. Intern. Conf. AIMC-1, Karlsruhe, Germany, August 3 – 6 (Ed. J.Calmet, J.A.Campbell). — 1992. N 737.
77. Ben-Ari M. Mathematical Logic for Computer Science. — Prentice Hall International (UK) Ltd. — 1993. — 305 p.
78. Gabbay D. Elementary Logic (A procedural perspective). — Prentice Hall Europe. — 1998. — 359 p.
79. Goldlatt R. Logics of Time and Computation // Lecture Notes. — Center for the Study of Language and Information. — 1987. — № 7.
80. Letichevsky A.A., Kapitonova J.V., Konozenko S.V. Computations in APS // Informatika'91. Franko-Sovietique Simposium: Theoretical Computer Science & Methods of Compilation and Program Construction. — Grenoble. — 1991 16 – 18 Oktobre — P. 133–152.
81. Post E.L. A variant of a recursive unsolvable problem // Bull. of Math. Soc. — 1946, — 52, N 4. — P. 264–268.
82. Revuz D. Minimization of acyclic deterministic automata in linear time Theoret. Comp. Sci. — 1992. — P. 181–189.
83. Downey P.J., Sethy., Tarjan .E. Variation on the common subexpression problem // J.ASM. — 1980. — P. 758–771.

ЗМІСТ

ПЕРЕДМОВА	3
Ч а с т и н а I. МАТЕМАТИЧНІ ОСНОВИ	5
Р о з д і л 1. МНОЖИНИ, ВІДНОШЕННЯ, КОМБІНАТОРИКА.6	
§ 1.1. <i>Мноожини, відношення, функції, операції</i>	6
1. Інтуїтивне поняття множини. Основні принципи	6
2. Операції над множинами. Закони для операцій	10
3. Декартовий добуток множин. Відношення	15
4. Приклади відношень	18
5. Приклади відображень	30
Контрольні питання	37
Задачі та вправи	38
§ 1.2. <i>Елементи комбінаторного аналізу</i>	42
1. Основне правило комбінаторики	42
2. Число різних k -елементних підмноожин n -елементної множини ..	43
3. Число підмноожин даної множини	46
4. Перестановки і розміщення упорядкованих множин	47
5. Перестановки з повтореннями	47
6. Розміщення елементів множини	49
7. Комбінації елементів з повтореннями	51
8. Біном Ньютона	52
9. Властивості біноміальних коефіцієнтів	54
10. Метод рекурентних співвідношень	55
11. Метод включень і вилучень	57
12. Метод продуктивних функцій	61
Контрольні питання	64
Задачі і вправи	64
Р о з д і л 2. ОСНОВНІ ПОНЯТТЯ ЗАГАЛЬНОЇ АЛГЕБРИ	67
§ 2.1. <i>Універсальні алгебри</i>	67
1. Загальні відомості	67
2. Відношення конгруентності	68
3. Гомоморфізми універсальних алгебр	69
4. Мова (алгебра) термів	72
5. Похідні операції і скінченні алгебри	75
§ 2.2. <i>Вільні алгебри та їх основні властивості</i>	76
1. Абсолютно вільні алгебри	76
2. Вільний групoid	77
3. Вільна напівгрупа	77

4. Вільна комутативна напівгрупа	79
5. Вільні групи	79
6. Вільні абелеві групи.....	83
7. Вільне кільце.....	88
8. Векторні простори	94
9. Булеві алгебри	96
10. Структури	107
11. Багатоосновні алгебри. Алгебра алгоритмів Глушкова.....	114
§ 2.3. Повні структури і напівкільця	117
1. Повні структури	117
2. Замкнуті напівкільця	119
Контрольні питання	121
Задачі і вправи	121
Розділ 3. ЕЛЕМЕНТИ ТЕОРІЇ АЛГОРИТМІВ І МАТЕМАТИЧНОЇ ЛОГІКИ	124
§ 3.1. Поняття алгоритму і алгоритмічної системи	124
1. Інтуїтивне поняття алгоритму	124
2. Обчислювані і частково рекурсивні функції. Теза Черча	125
3. Алгоритмічні проблеми	131
4. Машини Поста і машини Тьюрінга	133
5. Алгоритмічна система Маркова	138
Контрольні питання	152
Задачі і вправи	152
§ 3.2. Числення висловлювань	154
1. Алфавіт і формули	154
2. Інтерпретація формул логіки висловлювань.....	156
3. Повні системи зв'язок	157
4. Система аксіом для числення висловлювань.....	159
5. Теорема дедукції	162
6. Несуперечність і повнота числення висловлювань	165
7. Числення висловлювань і булеві алгебри	168
§ 3.3. Методи перевірки тотожності істинності формул числення висловлювань	169
1. Алгебраїчний метод	170
2. Метод Куайна	170
3. Метод редукції	171
4. Метод Девіса—Патнема.....	172
5. Метод резолюцій	174
Контрольні питання	176
Задачі і вправи	177
§ 3.4. Числення предикатів першого порядку	178
1. Алфавіт і формули	178
2. Істинність, інтерпретації, моделі	180
3. Аксіоматика і правила виведення	183
4. Основні властивості теорії першого порядку.....	185
5. Нормальні форми формул логіки предикатів першого порядку.	195
6. Скулемівські стандартні форми	198

7. Класифікація логік	200
8. Поняття про некласичні логіки	202
8.1. Пропозиційна модальна логіка.....	202
8.1.1. Синтаксис	202
8.1.2. Семантика.....	203
8.1.3. Аксіоматика ПМЛ	205
8.1.4. Логіка можливого і необхідного.....	206
8.1.5. Динамічна логіка.....	207
8.1.6. Властивості бінарних відношень і різновиди ПМЛ ..	207
8.2. Мультимодальні логіки	209
8.2.1. Синтаксис	209
8.2.2. Семантика.....	209
8.2.3. Аксіоматика ПММЛ	210
Контрольні питання	210
Задачі і вправи	210
8.3. Пропозиційна темпоральна логіка	211
8.3.1. Синтаксис	211
8.3.2. Семантика.....	211
8.3.3. Аксіоматика ЛПТЛ.....	212
Контрольні питання	214
Задачі і вправи	214
8.4. Пропозиційна n -значна логіка	214
Контрольні питання	217
Задачі і вправи	217
9. Поняття алгебаїчної системи	218
Контрольні питання	221
Задачі і вправи	221
Розділ 4. ЕЛЕМЕНТИ ТЕОРІЇ ГРАФІВ	224
<i>§ 4.1. Означення графів, різновиди графів</i>	224
1. Означення неорієтованого графа	224
2. Різновиди графів.....	226
3. Ізоморфізм графів. Підграфи.....	229
Контрольні питання	230
Задачі і вправи	230
<i>§ 4.2. Операції над графами</i>	230
1. Операція вилучення ребра	230
2. Операція вилучення вершини	231
3. Операція введення ребра.....	232
4. Операція введення вершини в ребро	232
5. Операція об'єднання графів.....	232
6. Добуток графів.....	233
7. Ототожнення (злиття) вершин	233
8. Операція стягування ребра	233
9. Операція роздвоєння (розщеплення) вершини	234
10. Операція з'єднання графів	234
11. Операція доповнення графа.....	235
<i>§ 4.3. Властивості графів</i>	236
1. Маршрути, цикли, зв'язність.....	236

2. Властивості регулярних графів	237
3. Властивості двочастинних графів	238
4. Властивості зв'язних графів	239
5. Метричні характеристики зв'язних графів.....	243
6. Властивості ейлерових графів	244
7. Властивості гамільтонових графів	246
§ 4.4. Матриці і графи	247
1. Матриці суміжностей і досяжності.....	247
2. Матриця Кірхгофа	250
3. Матриця інцидентності графа	251
§ 4.5. Планарність і укладання графів	251
1. Плоскі і планарні графи.....	251
2. Грані плоского графа. Формула Ейлера	257
§ 4.6. Розфарбування графів. Хроматичне число	259
1. Правильне розфарбування	259
2. Практичні задачі, що зводяться до задачі розфарбування	260
3. Хроматичні числа деяких графів	261
4. Гіпотеза чотирьох фарб	263
5. Гіпотеза чотирьох фарб для карт	264
§ 4.7. Нескінченні графи	265
Контрольні питання	268
Задачі і вправи	268
§ 4.8. Дерева	269
1. Означення дерева. Властивості дерев.....	269
2. Фундаментальна система циклів графа.....	273
3. Остов найменшої ваги.....	275
§ 4.9. Орієнтовані графи і дерева.....	276
1. Основні поняття	276
2. Бінарні відношення і орграфи	279
3. Позначені графи і зображення термів.....	280
Контрольні питання	283
Задачі і вправи	283
Ч а с т и н а II. МАТЕМАТИЧНІ МОДЕЛІ	285
Р о з д і л 5. АБСТРАКТНА ТЕОРІЯ АВТОМАТІВ.....	286
§ 5.1. Основні означення	286
1. Означення автомата і його різновиди	286
2. Таблиці переходів і виходів.....	289
3. Граф переходів і виходів	289
4. Підавтомати. Гомоморфізми автоматів	292
5. Теорема про приведений автомат	296
Контрольні питання	299
§ 5.2. Представлення подій в автоматах. Автомати Мура і недетерміновані автомати	299
1. Автоматні відображення.....	300
2. Автоматні системи подій.....	303
3. Автомати Мура	305

4. Недетерміновані автомати.....	309
Задачі та вправи	311
Р о з д і л 6. ТЕОРИЯ СКІНЧЕННИХ АВТОМАТІВ.....	313
<i>§ 6.1. Події, алгебра регулярних подій. Основна теорема теорії скінчених автоматів</i>	<i>313</i>
1. Регулярні події	313
2. Теорема аналізу скінчених автоматів	315
3. Теорема синтезу скінчених автоматів	319
Задачі та вправи	327
<i>§ 6.2. Практичні методи аналізу і синтезу скінчених автоматів</i>	<i>328</i>
1. Рівняння в алгебрі подій	328
2. Системи лінійних рівнянь	330
3. Застосування рівнянь в алгебрі подій до задач аналізу і синтезу скінчених автоматів	331
4. Основний алгоритм синтезу скінчених автоматів	333
Контрольні питання	338
Задачі та вправи	338
<i>§ 6.3. Мінімізація скінчених автоматів без виходів</i>	<i>339</i>
1. Загальний алгоритм мінімізації	339
2. Алгоритм мінімізації скінчених автоматів без виходів	341
3. Алгоритм мінімізації ациклічних автоматів	342
<i>§ 6.4. Алгоритми побудови конгруентних замикань для скінчених автоматів</i>	<i>348</i>
1. Означення відношення конгруентного замикання	348
2. Загальний алгоритм обчислення конгруентного і симетричного конгруентного замикань	350
3. Алгоритм обчислення конгруентних замикань для ациклічних автоматів	357
Контрольні питання	360
Задачі і вправи	360
Р о з д і л 7. МОДЕЛІ АЛГОРИТМІВ І ПРОГРАМ	362
<i>§ 7.1. Машини Тьюрінга</i>	<i>362</i>
1. Означення і функціонування	362
2. Словесне представлення машини Тьюрінга	365
3. Алгоритмічно розв'язні і нерозв'язні проблеми	366
Контрольні питання	370
Задачі і вправи	370
<i>§ 7.2. Скінченні автомати з однією стрічкою</i>	<i>371</i>
1. Означення автомата з однією стрічкою	371
2. Проблема пустоти	372
3. Проблема нескінченності	372
<i>§ 7.3. Магазинні автомати</i>	<i>374</i>
1. Означення магазинного автомата	374
2. Представлення мов в магазинних автоматах	376
3. Аналіз магазинних автоматів	377
4. Синтез магазинних автоматів	379

Контрольні питання	381
Задачі і вправи	381
§ 7.4. Дискретні динамічні системи	382
Контрольні питання	385
Задачі і вправи	385
§ 7.5. U-Y-схеми програм над пам'яттю	385
Контрольні питання	389
Задачі і вправи	389
Розділ 8. ФОРМАЛЬНІ ГРАМАТИКИ І ФОРМАЛЬНІ МОВИ	390
§ 8.1. Регулярні граматики та їх властивості	390
1. Означення формальної граматики	390
2. Класифікація граматик	395
3. Праволінійні і ліволінійні граматики (регулярні граматики) і скінченні automati	398
Задачі і вправи	402
§ 8.2. Контекстно-вільні граматики та їх властивості	403
1. Контекстно-вільні граматики і рівняння	403
2. Рівняння, кв-мови і магазинні automati	408
Задачі і вправи	410
Частина III. ЗАСТОСУВАННЯ	411
Розділ 9. АЛГЕБРИ В КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЯХ	412
§ 9.1. Перетворення у векторних просторах	412
1. Переміщення площини	412
2. Групи перетворень площини	417
3. Переміщення простору	418
4. Групи перетворень простору	423
5. Масштаб	426
6. Проекції	427
§ 9.2. Реляційна алгебра і поняття реляційної бази даних	429
1. Означення реляційної алгебри	429
2. Деякі властивості операцій реляційної алгебри	435
Контрольні питання	437
Задачі і вправи	437
§ 9.3. Алгебраїчна система спискових структур	438
1. Списки. Операції над списками	438
2. Реалізація списків у пам'яті ЕОМ	444
3. Складність виконання операцій над списками	445
4. Різновиди списків	446
5. Задання графів за допомогою списків	447
6. Задання множин і термів за допомогою списків	448
Контрольні питання	450
Задачі і вправи	450

Р о з д і л 10. ТЕОРІЯ АВТОМАТІВ І ГРАФІВ У КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЯХ	452
§ 10.1. Універсальний програмний автомат.....	452
§ 10.2. Застосування в системах підготовки та редакції текстів.....	458
1. Ідентифікація слів при побудові текстових редакторів	458
2. Поняття гіпертексту	463
Контрольні питання	464
Задачі і вправи	464
§ 10.3. Комп'ютерна алгебра.....	464
1. Задача спрощення алгебраїчних виразів	464
2. Важливі спрощення	467
3. Повні системи редукцій, критичні пари, алгоритм поповнення	468
4. Алгоритм Кнута–Бендікса критичної пари	471
5. Алгоритм поповнення Кнута–Бендікса	473
6. Проблема загальних підвіrazів і канонізації для вільних груп	475
6.1. Задання термів за допомогою скінчених автоматів	475
6.2. Задання скінчених автоматів за допомогою системи	
правил переписування	480
6.3. Задачі з теорії скінченнопороджених вільних груп	481
6.4. Скінченні алгебри	485
Задачі і вправи	488
§ 10.4. Теорема Холла та її застосування	489
1. Трансверсалі.....	491
2. Латинські квадрати	492
§ 10.5. Теорема Менгера і потоки в мережах.....	493
Задачі і вправи	501
Р о з д і л 11. МЕТОДИ ПОШУКУ ДОВЕДЕННЯ ТЕОРЕМ У ЛОГІЦІ ПРЕДИКАТИВ.....	502
§ 11.1. Метод Ербрана	503
1. Основні означення	503
2. Семантичні дерева	507
3. Теорема Ербрана.....	508
§ 11.2. Метод резолюцій.....	510
1. Контрарні пари і підстановки	510
2. Алгоритми уніфікації.....	512
3. Метод резолюцій для логіки предикатів	
першого порядку (ЛППП)	516
4. Приклади застосування методу резолюцій	521
5. Правило поглинання	523
Контрольні питання	527
Задачі і вправи	527
Р о з д і л 12. ОСНОВНІ ПОНЯТТЯ ТЕОРИЇ ПРОГРАМНИХ ІНВАРІАНТІВ	529
§ 12.1. Інваріанти в станах U-Y-програм і мові інваріантів.....	529
1. Означення програмного інваріанта	529
2. Методи пошуку програмних інваріантів	532

<i>§ 12.2. Мова рівностей. Основні задачі</i>	534
1. Основні задачі теорії програмних інваріантів	534
2. Алгоритми пошуку програмних інваріантів.....	536
<i>§ 12.3. Пошук інваріантів у програмах над вільними алгебрами даних</i>	544
1. Абсолютно вільні алгебри	544
<i>§ 12.4. Векторні простори</i>	553
1. Розв'язки основних задач.....	553
2. Вільні абелеві групи і комутативні напівгрупи.....	558
3. Приклади.....	559
Задачі і вправи	566
СПИСОК ЛІТЕРАТУРИ	568

Наукове видання

**КАПІТОНОВА Юлія Володимирівна
КРИВИЙ Сергій Лук'янович
ЛЕТИЧЕВСЬКИЙ Олександр Адольфович
ЛУЦЬКИЙ Георгій Михайлович
ПЕЧУРІН Микола Капітонович**

**ОСНОВИ
ДИСКРЕТНОЇ
МАТЕМАТИКИ**

Київ, видавництво “Наукова думка”

Оформлення художника *Н.М. Абрамової*
Художній редактор *І.М. Галушка*
Технічний редактор *С.Г. Максимова*
Коректор *Л.Г. Бузіашвілі*
Комп’ютерна верстка *Л.Є. Ляскalo*

Підп. до друку 26.11.2001. Формат 60x90/16.
Папір офс. №1. Гарн. Таймс. Офс. друк.
Ум.-друк.арк. 36,5. Ум. фарбо-відб. 36,5.
Обл.-вид. арк. 50,0. Тираж 3000 прим. Зам. 1-324.

Видавництво “Наукова думка”
Р.с. №05417561 від 16.03.95
01601 Київ 1, вул. Терещенківська,3.

ВАТ “Книжкова друкарня наукової книги”
04107, Київ 107, вул. Багговутівська,17—21