МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДНІПРОПЕТРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ЗАЛІЗНИЧНОГО ТРАНСПОРТУ ІМЕНІ АКАДЕМІКА В. ЛАЗАРЯНА

ДИСКРЕТНІ ТА АЛГОРИТМІЧНІ СТРУКТУРИ В ІНСТРУМЕНТАРІЇ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

НАВЧАЛЬНИЙ ПОСІБНИК

Дніпропетровськ
2016

УДК 004.94 (075.8):510.51
ББК 32.97я7
Д48

Автори: доктор технічних наук, професор В. В. Скалозуб,
кандидат фізико-математичних наук, доцент В. М. Ільман,
кандидат технічних наук, доцент Ю. М. Івченко,
кандидат технічних наук, доцент В. О. Андрющенко

Рецензенти: В. Є. Білозьоров, доктор фізико-математичних наук, професор (ДНУ),
О. І. Міхальов, доктор технічних наук, професор (НМА України).

Рекомендовано до друку вченою радою Дніпропетровського національного університету залізничного транспорту імені академіка В. Лазаряна як навчальний посібник (протокол № 9 від 24.04.2016). Зареєстровано НМВ університету (реєстр. № 265/16-3 від 18.05.2016)

УДК 004.94 (075.8):510.51
Дискретні та алгоритмічні структури в інструментарії програмної інженерії [Текст] : навч. посіб. / В. В. Скалозуб, В. М. Ільман, Ю. М. Івченко, В. О. Андрющенко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Дніпропетровськ, 2016. – 254 с.
ISBN 978-966-8471-73-5

Викладено основні положення комп'ютерної математики, формальних конструктивних граматичних і алгоритмічних структур, методи оцінки характеристик алгоритмів, прикладні питання трансляторів, теорії графів, мереж Петрі, що призначені для розвитку навичок структуризації предметних областей програмування, створення конструктивних об'єктів та їх застосування при моделюванні завдань програмної інженерії.

Для студентів, викладачів і фахівців у галузі інформатики та програмної інженерії.

Іл. 42.Табл.11. Бібліогр. : 110 назв.

ISBN 978-966-8471-73-5

ВСТУП

Розвиток і досягнення науки в теоретичних і прикладних галузях інформатики, розробка нових технологій переробки інформації (мережі передачі даних, розподіл інформаційних ресурсів, тощо), існування широкого спектру мов програмування (Java, C#, C++, Prolog, Turbo Pascal Pithon, SQL) з різними ідеологіями представлення предметних областей, вимагають від розробників глибоких знань сфери інформаційних систем.

Сучасний фахівець із програмної інженерії повинен володіти різноманітними знаннями з багатьох наук (дивися «Рекомендации по преподаванию програмной инженерии, информатики в университетах» www.intuit.ru Москва, 2007; в подальшому - «Положення»). В першу чергу - це математична логіка, теорії класичних та конструктивних множин, відношень, автоматів, алгоритмів і багато іншого.

Метою написання посібника є розвиток навичок володіння математичним інструментарієм, необхідних для структуризації предметних областей і створення конструктивних об’єктів, а також їх застосування при моделюванні завдань програмної інженерії.

Викладання матеріалу першої частини починається з попередніх математичних відомостей, розгляду та застосування формальних конструктивних граматичних і абстрактних структур та їх застосування. Також у першому розділі розглянуто елементи дискретної математики: формалізми та формальні системи, логічні системи, класичні та конструктивні множини, деякі питання відношень, графи і автомати.

В другу частину включені розділи алгоритмічних структур, методи виміру і оцінки алгоритмів, прикладні питання трансляторів, теорії графів, мереж Петрі.

Комп’ютерна математика є складовою програмування, містить такі дисципліни, як теорії формальних граматик, автоматів, графів та алгоритмів, необхідних для набуття навичок розробки і програм. Також у першому розділі розглянуто елементи дискретної математики:

формалізми та формальні системи, логічні системи, класичні та конструктивні множини, деякі питання відношень, графів і автоматів.

У прикладних завданнях програмної інженерії важливим є конструювання штучних об'єктів (розробка алгоритмів, програм), а також їх дослідження. У наступному розділі посібника розглядаються інші складні математичні об'єкти: автомати, алгебри, формальні структури. Алгебри призначені для дослідження операцій з елементами, на яких вони визначені. В роботі розглянуті класичні алгебри і частково системи конструктивних алгебр Глушкова.

Через те що, за межами визначення алгебраїчних структур, алгебраїчних систем та систем алгоритмічних алгебр знаходяться правила виконання операцій сигнатур і аксіоми застосування цих правил, для „повноцінного" дослідження мов і мовних конструкцій необхідно застосовувати різні математичні розділи: теорію формальних систем та алгебр.

В посібнику представлено математичний об'єкт *формальна структура*, в межах якої для заданої предметної області можливо створювати мовні, алгоритмічні та інші конструкції, а також проводити їх дослідження.

Посібник написаний за матеріалами лекцій, які читалися в різний час студентам (програмістам, прикладним математикам, кібернетикам) на факультеті технічної кібернетики Дніпропетровського національного університету залізничного транспорту імені академіка В.Лазаряна.

Доведення тверджень виділені символами $\triangleright$ – початок і $\triangleleft$ – завершення доведення. В тексті смислові нововведення виділені курсивом.

Як правило, підрозділи підручника завершуються задачами та вправами. Для зручності кожен розділ посібника супроводжується списком літератури, кожна частина має додатки з прикладами вирішення завдань.

Посібник в основному призначений для студентів, які навчаються за фахом інженерія програмного забезпечення, але його матеріали можуть використовуватися студентами і викладачами інших напрямків інформаційної галузі.

Автори висловлюють щиру подяку колективу кафедри «Комп'ютерні інформаційні технології» за постійну увагу і підтримку цієї роботи.

Розділ 1.
Завдання програмної інженерії. Формальний інструментарій

Розділ присвячений початковим відомостям з теорії формальних систем необхідним для подальшого викладення та моделювання елементів програмної інженерії. Розглянуто і наведено поняття питань: *формалізм*, *формальна система*, *формальна структура*, *висловлювання*, *логічні числення*, *числення предикатів*, *логічні перетворення*.

1.1. Мета розділу

Як відомо набуття людством знань відбувається поступово. Це стосується і наукових напрямків діяльності людини. Наприклад, математика бере початок від природних систем числення, геометрії, платонівських класифікацій, логіки Аристотеля, вводяться елементи множин. При чому вважалося, що математична будівля «міцна і непохитна» доти поки в рамках теорії множин не були виявлені парадокси. (Відносно різних парадоксів і проблем розвитку математики дивися, наприклад, розділ «Основи математики» роботи С. Кліні [3]). Для підкріплення фундаменту математики була створена теорія математичної логіки [5, 6], для якої в свою чергу «сконструйована» теорія формальних систем. Формальні системи [7, 8] є зручним інструментом при відтворені деяких теорій в штучних системах таких, як теорія формальних граматик, теорія алгоритмів, автоматів і інших, які є основним інструментарієм прикладної науки «Програмна інженерія». Тому цей розділ присвячений початковим відомостям необхідним для подальшого викладення та моделювання елементів програмної інженерії.

Отже метою розділу є наведення елементів формальних теорій: логіки висловлювань, числення висловлювань і предикатів, як основи інструментарію програмної інженерії.

1.2. Задачі програмної інженерії

Програмна інженерія розділ людських знань і прикладної методології розробки та створення штучних об'єктів і систем, незавершена у своєму формуванні та визначені. Визначення програмної інженерії досить різноманітні. Для нас більш прийнятним є визначення наведене у «Положеннях», яке виглядає так.

Визначення 1.1. Програмна інженерія є такою формою інженерії, яка застосовує принципи інформатики і математики для отримання рентабельних рішень програмного забезпечення.

Причому під «рентабельністю рішень» розуміється оцінка і аналіз станів життєвого циклу програмного забезпечення: специфікації, проектування, розробки та еволюції і ін.

Програмна інженерія розв'язує задачі

- дискретної природи,
- інтегрує концепції математики і інформатики з інженерними підходами до відтворення матеріальних цінностей,
- розробки системних моделей,
- розробки надійних методів створення якісного програмного забезпечення.

Таким чином, розглянуті у визначенні 1.1 математичні принципи програмної інженерії охоплюються, в цьому посібникові, наступною схемою (рис. 1.1).

1.3. Елементи формалізмів

В процесі набуття знань людина використовує ті чи інші узагальнення об'єктів через позначки, тобто вводе *формальні* позначення предметів, подій, явищ тощо. Цей прийом має перевагу в тому, що дозволяє розглядати пізнавальну проблему як формалізовану узагальнену та без несуттєвих конкретних подробиць і досліджувати її

за законами формальних теорій. В подальшому будуть розглянуті деякі формалізми конструктивної математики.

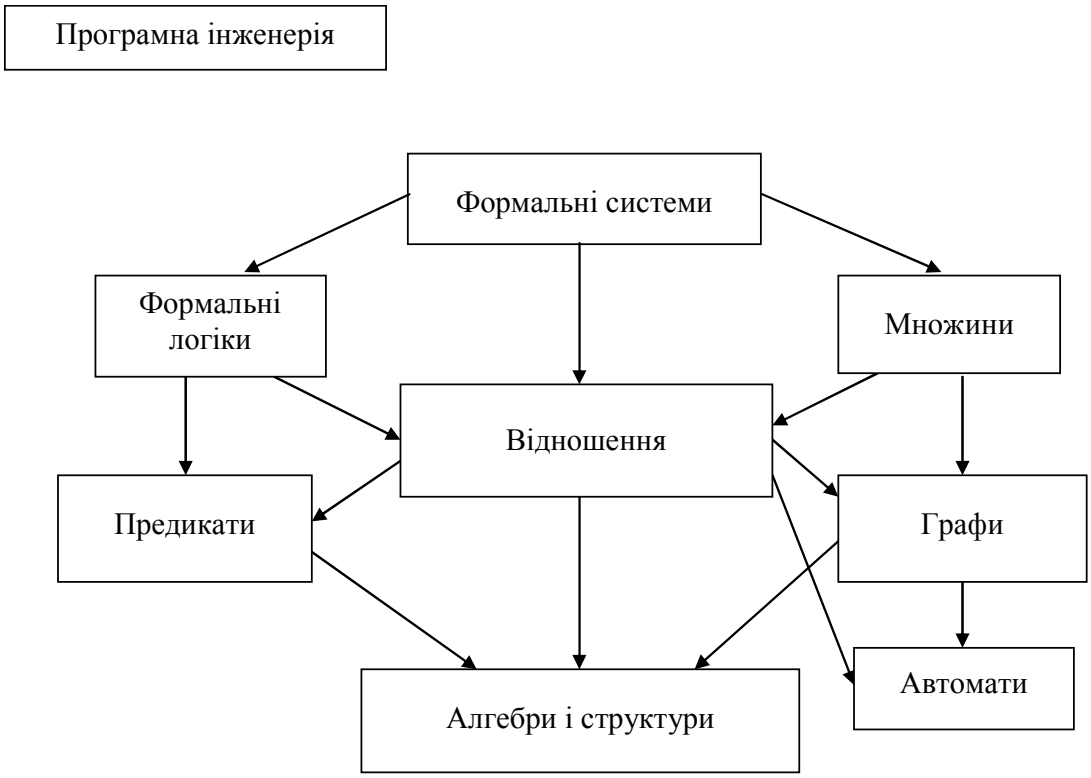


Рис. 1.1. Математичні принципи програмної інженерії

1.3.1. Загальні поняття

Розглянемо не формальний приклад запису. «На деякій лінії певної довжини побудовано квадрат, до суміжних сторін якого добудовано два однакові прямокутники так, що одна із сторін кожного прямокутника співпадає зі стороною квадрату, а кожна з інших їх сторін має за довжину число 5, і загальна площа побудованої фігури є число 39». Зрозуміло, що з'ясувати змістовну суттєвість цього запису зразу не просто. Але, якщо прийняти довжину лінії за x, тоді для площі геометричної фігури отримаємо *формальну* форму запису, запропонованого ще Ал-Хорезмі [2]

$$x^2 + 10x = 39 \qquad (1.1)$$

В залежності від змісту запис (1.1) можливо назвати *рівнянням, формулою*, ліву його частину – *виразом* (*термом*). Відомо, що формальне позначення може бути *константою* , якщо його значення не змінне, тобто є конкретне значення $a = 10$, $b = 39$ або *змінним зна-*

ченням в протилежному випадку – так позначення x є змінною величиною. Можливо переписати вираз (1.1) в *загальному формальному* вигляді

$$x^2 + ax = b \tag{1.2}$$

Введення знаків, крім скорочення записів, дозволяє формалізувати властивості дій, формалізувати *алгебраїчні перетворення*. Наприклад, замість твердження «від зміни місць множників їх добуток не змінюється» можливо формально записати $x \cdot y = y \cdot x$ або формалізувати правила перетворення виразу $(x+y)^2$

$$(x+y)^2 = (x+y)(x+y) = x(x+y) + y(x+y) = \\ = x^2 + xy + yx + y^2 = x^2 + 2xy + y^2$$

Слід звернути увагу на те, що в виразі добутку знак множення $(\cdot)$, як правило не пишеться, але є ще одна дія $\boxtimes$ – *конкатенація* [1], за якою до одного об'єкту послідовно приписується інший об'єкт, у виразі з якою знак $\boxtimes$ не пишеться, тобто $x \boxtimes y = xy$. Проблема розпізнання об'єкту xy розв'язується змістовно. Результат дії конкатенації звуть *ланцюжком*, *словом*, *лексемою*, *ім'ям* та інше. Так, якщо ввести у розгляд скінчену сукупність об'єктів-позначок, яку за звичаєм назвемо *алфавітом* і позначимо його буквою A, тоді дія конкатенації над алфавітом створить сукупність формальних ланцюжків A^*. Сукупність ланцюжків (включаючи і порожній ланцюжок) A^* зветься *вільною формальною мовою*. Очевидно, для дії конкатенації мають місце формули $x \boxtimes y \neq y \boxtimes x$, $(x \boxtimes y) \boxtimes z = x \boxtimes (y \boxtimes z)$.

1.3.2. Формальні системи

На заданих вільних формальних мовах можна побудувати більш складні формалізми, наприклад, *формальні системи* [1, 4]. Конструктивно такі системи можливо створити за наступною схемою:

1) на заданому алфавіті будується формальна мова;
2) в цій мові виділяється набір ланцюжків (формул), яка приймається за *аксіоми* (*початок виводу*);
3) задаються *формальні правила виводу*, за допомогою яких виводяться (конструюються) ланцюжки (формули) і інше.

Нехай задано алфавіт A, над яким побудована формальна мова A^*, далі припустимо, що Λ аксіоми в ній і P – задана сукупність правил виводу з аксіом Λ, тоді формальну систему над алфавітом A визначимо як

Визначення 1.2. Формальною системою $\mathbb{S}_A$ над алфавітом A назвемо упорядковану четвірку $\mathbb{S}_A = \langle A, A^*, \Lambda, P \rangle$.

Для опису складових формальної системи завжди використовується зрозуміла людині спеціальна мова. Така зовнішня по відношенню до системи мова зветься *метамовою* або *мовою дослідника* [3]. Метамовою може бути звичайна людська мова або формалізована (*штучна*) мова, створена людиною. Наприклад, в нашому випадку, посібник написаний на українській мові, тому метамовою є українська мова. А дослідження формальних систем виконуються за допомогою формальних *предметних мов*. Так, для нашого випадку, до предметної мови включено: алфавіти, ланцюжки, аксіоми, правила виводу, тощо.

Аксіоми формальної системи можуть бути безпосередньо ланцюжками з сукупності A^*, що визначається за певними правилами. Для розпізнання аксіом будемо позначати їх грецькою буквою σ або $\sigma_1, \sigma_2, \ldots$. Форма запису правил P може бути будь якою, наприклад, типу синтаксичних діаграм або діаграм Бекуса-Наура як в програмуванні, або канонічними системами Поста [7] та інше. Але частіше ми будемо користуватися більш компактною формою *заміщень* (операція *підстановки*) $x \to y$, яка читається «*якщо* x, *тоді* y» або так «y *виводиться з* x», або «x *заміщується* y». Завжди будемо вважати, що сукупність правил виводу P є скінченою, тобто складається з скінченої кількості правил.

Визначення 1.3. Ланцюжок зветься *виведеним* з аксіом Λ формальної системи, якщо існує послідовність правил (*виводів*), у тому числи однакових, за якою він виводиться при застосуванні цих правил.

Так, наприклад, вивід ланцюжка $x = a_1 a_2 a_3$ з аксіоми σ за допомогою правил: $p_1 : \sigma \to a_1$, $p_2 : a_1 \to a_1 a_2$, $p_3 : a_2 \to a_2 a_3$; записується у вигляді $\sigma \xrightarrow{p_1} a_1 \xrightarrow{p_2} a_1 a_2 \xrightarrow{p_3} a_1 a_2 a_3$. Цей вивід можливо представити також через вивід в системі Поста (*числення Поста*).

Припустимо, що задана сукупність змінних X (змінних об'єктів) і скінчений алфавіт A, при чому позначення об'єктів алфавіту A і змінних різні. На цих сукупностях створимо наступні дві конструкції.

Терм – це будь який ланцюжок (формула) t побудований за допомогою конкатенації або інших операцій над об'єктами сукупностей A і X.

Схема – це конструкція з термів $t, t_1, t_2, \ldots, t_i$; $(i \geq 0)$, яка записується у вигляді

$$C_i = \frac{t_1, t_2, \ldots, t_i}{t}.$$

тут терми $t_1, t_2, \ldots, t_i$ – є *посиланнями*, а t – є *висновком схеми.*

Схема без посилань зветься *аксіомою.*

Канонічну систему Поста утворює упорядкована четвірка $\langle A, X, \Lambda, C \rangle$, в якій Λ - система аксіом утворених на словах алфавіту A (схеми без посилань), C - скінчена сукупність схем C_i.

Зрозуміло, що канонічна система утворює формальну систему $\mathbb{S}_B$, якщо алфавіт B є сукупність алфавітів A та X і B^* – формальна мова над B.

Кожна з схем C_i сукупності C задає вивід терму t за зв'язаною скінченою послідовністю формул $t_1, t_2, \ldots, t_i$, в якій $t_i = t$, а t_j, $j = 1, 2, \ldots, i-1$ – або аксіома, або одне з правил (схеми) виводу формальної системи $\mathbb{S}$, або їх наслідок.

Для вище розглянутого прикладу схема виводу в системі Поста має вигляд

$$\frac{\sigma, p_1, p_2, p_3}{x},$$

де терм x є наслідок застосування схеми (правила)

$$p_3 : \frac{a_2, a_2 a_3}{a_2 a_3}$$

до результату застосування схем

$$p_1 : \frac{\sigma, a_1}{a_1} \text{ і } p_2 : \frac{a_1, a_1 a_2}{a_1 a_2}.$$

Більш повне знайомство з канонічними системами Поста та їх застосуванням можливо отримати за роботою [3].

1.3.3. Поняття формальної структури

Визначення 1.4. Послідовність виводів ланцюжка визначає його структуру (теорію виводу), тобто його будову.

Визначення 1.5. Повна сукупність виведених ланцюжків $\mathbb{L}$ в формальній системі $\mathbb{S}$ зветься *формальною мовою* виведену в формальній системі і записується так $\Lambda \xrightarrow[\mathbb{S}]{} \mathbb{L}$.

Очевидно, що формальна мова $\mathbb{L}$ виведена в формальній системі $\mathbb{S}_A$ є частина вільної мови побудованої над алфавітом A.

Дамо пояснення цих понять на наступному прикладі.

Приклад 1.1. Необхідно утворити формальну мову у системі $\mathbb{S}$ над сукупністю об'єктів $A=\{a,b,c\}$ за аксіомами Λ:

$$1)\ \sigma = a,\ \ 2)\ \sigma = b,\ \ 3)\ \sigma = c$$

та правилами виводу

$$P:\ 1)\ \sigma \to a\sigma a\,,\ \ 2)\ \sigma \to b\sigma b\,.$$

Зрозуміло, що сукупність A^* складається з будь яких ланцюжків побудованих за допомогою конкатенації над заданими об'єктами алфавіту A. Об'єктами ж мови $\mathbb{L}$, створеними цією системою будуть симетричні ланцюжки, наприклад, ланцюжок $aabcbaa$, який отримаємо в результаті послідовності виводів

$$\sigma \xrightarrow{1} a\sigma a \xrightarrow{1} aa\sigma aa \xrightarrow{2} aab\sigma baa \xrightarrow{3} aabcbaa$$

за допомогою подвійного застосування першого правила потім другого правила та третьої аксіоми формальної системи.

Формальна мова для цього випадку має вигляд нескінченої сукупності ланцюжків

$$\mathbb{L}=\{a,b,c,aaa,aba,aba,bab,bbb,bcb,aaaaa,aabaa,$$
$$aacaa,ababa,abbba,abcba,\ldots\}\,.$$

В подальшому ми розглянемо елементи формальної системи логіки, формальної системи множин та інші системи.

1.3.4. Завдання і вправи

1. Побудувати алфавіт дисциплін, які вивчаються студентами. Що означає при цьому порожній елемент алфавіту?
2. Яка вільна мова утворюється на алфавіті:

   а) $A=\{a\}$;

   б) $A=\{a,b\}$;

   в) $A=\{a,b,c\}$?
3. Які терми можливо побудувати на алфавіті змінних X та алфавіті знаків Z:

   а) $X=\{x,y,z\}$, $Z=\{+,-\}$;

   б) $X=\{a,x,y\}$, $Z=\{\boxtimes,+,\cdot\}$;

   в) $X=\{a,b,c\}$, $Z=\{\cdot,/\}$?
4. Задати правила виводів ланцюжків на алфавітах вправи 3 з аксіоми $\sigma=u$, де u один з об'єктів алфавіту X.
5. Записати схеми з посиланнями термів вправи 4.
6. Записати канонічні системи Поста для виводів вправи 4.
7. Утворити формальні системи над сукупностями алфавітів вправи 3 і аксіомами над алфавітами А цієї вправи.
8. Які формальні мови утворюють формальні системи вправи 7?
9. Визначити формальну структуру ланцюжків формальних мов завдання 8.
10. Побудувати формальну систему з операціями додавання та віднімання над цілими числами.
11. Побудувати формальну систему з операціями множення і ділення над раціональними числами.
12. Побудувати формальну систему на зв'язках сімейних відношень.
13. Спробувати записати формальну систему знаходження межі елементарних виразів $\lim\limits_{x\to 0} f(x)$ та формальну систему звичайної похідної $\dfrac{df(x)}{dx}$.

1.4. Формальні логіки

У цьому підрозділі розглянуто об'єкти висловлювань та показано, як на їх основі будується формальна логічна система [4, 5, 6]. Предметна мова формальних логік складається з понять: *висловлювання, логічний алфавіт, логічні зв'язки, логічні формули* та інших понять.

1.4.1. Висловлювання та дії над ними

Первинним поняттям *логіки* (вчення про формальне мислення) є висловлювання. Як правило, первинні поняття вводиться описово, через пояснення. Таким же чином діють і з логічними висловлюваннями. Під *логічним висловлюванням* слід розуміти оповідальне речення у деякій метамові, відносно якого можливо сказати чи воно є *істинним*, чи *хибним* або *невизначеним*. Наприклад, «число 25 поділяється на 5», «швидкість літака більша за швидкість воза», «для будь якої дійсної змінної x виконується умова $x^2 \geq 0$» є істинними висловлюваннями, а висловлювання «$2 \neq 2$», «для будь якої дійсної змінної x має місце $x = x+1$» є хибними. Окрім таких *постійних висловлювань* можуть зустрічатися і висловлювання «$2x > 2$», істинність чи хибність якого визначається тим, яке значення набуває змінна x. Такі висловлювання звуться *змінними висловлюваннями*. Відволікаючись від природи висловлювань будемо їх називати *логічними об'єктами* або просто об'єктами.

Для логічних об'єктів введемо позначення за допомогою літер грецького алфавіту $\alpha, \beta, \delta, \cdots$. Скінчена сукупність логічних об'єктів утворює *логічний алфавіт*, який будемо позначати великими літерами, тобто $A = \{\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_k\}$. І введемо тернарний числовий алфавіт $T = \{0, \mu, 1\}$ значень логічних об'єктів. Так, якщо об'єкт α має істинне значення, будемо записувати $V(\alpha) = 1$, де V – початкова літера англійського слова «value» – величина; якщо – хибне значення тоді $V(\alpha) = 0$, у невизначеному випадку істинності (змінному висловлюванні) – $V(\alpha) = \mu$. Інколи замість записів $V(\alpha_1) = 1$, $V(\alpha_2) = 0$ або $V(\alpha_3) = \mu$, , якщо це не викликатиме непорозуміння, будемо скорочено записувати відповідно $\alpha_1 = 1$, $\alpha_2 = 0$, $\alpha_3 = \mu$ Очевидно, що в наших позначеннях величина істинності μ задовольняє умові

$0 < \mu < 1$. Таким чином алфавіт T повністю характеризує логічні об'єкти формальної системи.

Наведені вище об'єкти будемо називати *елементарними* або *атомарними*, тобто вони відповідають простим висловлюванням.

1.4.2. Формули висловлювань

З простих висловлювань можливо створити більш складні висловлювання за допомогою речових *зв'язок:* «і», «або», «ні», «якщо», «тоді», «тоді і тільки тоді, коли». Для цих зв'язок введемо відповідні позначення $\{\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow\} = Z$. Алфавіт Z звуть *алфавітом пропозиціональних зв'язок*. Тоді, наприклад, висловлювання: «(число π більше трьох) і (число π менше за число 4)», «якщо (трикутник прямокутний), тоді (сума квадратів його катетів дорівнює квадрату гіпотенузи)» можливо записати у вигляді наступних *формул*: $\alpha \wedge \beta$ і $\delta \Rightarrow \gamma$, де через $\alpha, \beta, \delta, \gamma$ позначені прості висловлювання виділені у дужках.

Взагалі то, складні об'єкти називають *формулами логіки висловлювань*. Наприклад, $(\alpha \wedge \beta) \vee \gamma$; $\alpha \wedge (\beta \vee \gamma)$; $(\alpha \wedge \overline{\beta}) \Rightarrow \gamma$; $(\alpha \Rightarrow \overline{\beta}) \Leftrightarrow (\delta \wedge \gamma)$ – є формули логіки висловлювань. В запису формул використовуються круглі дужки для виділення послідовності застосування зв'язок до об'єктів. Зрозуміло, що сукупність логічних формул утворюється над двома алфавітами: алфавітом об'єктів A та алфавітом зв'язок Z, тобто ця сукупність є $\{A, Z\}^*$. Для повноти формул до сукупності $\{A, Z\}^*$ також віднесемо прості об'єкти, щоб підкреслити ту обставину, що з простих об'єктів будуються всі формули логіки висловлювань. Якщо тепер в сукупності $\{A, Z\}^*$ виділити аксіоми Λ (деякі з них розглянемо в наступних пунктах) і ввести сукупність правил P, побудованих над $\{A, Z\}^*$, тоді і отримаємо формальну логічну систему $\mathbb{S}_A$. Наприклад, правила можуть мати вигляд заміщень:

$$P = \{1)\ \alpha \to \alpha \wedge \beta;\ \ 2)\beta \to (\alpha \vee \alpha_1);\ \ 3)\beta \vee \alpha_1 \to \gamma, \ldots\}.$$

Також, як і прості висловлювання, формули сукупності $\{A, Z\}^*$ є логічними об'єктами, тобто їх значення визначаються над числовим

алфавітом T. Так значення формул на основі підалфавіту зв'язок $\{\wedge,\vee,\neg\}$ над об'єктами α і β визначаються табл. 1.1 [1].

Таблиця 1.1

Значення логічних формул за зв'язками $\{\wedge,\vee,\neg\}$

$\alpha\wedge\beta$	0	μ	1	$\alpha\vee\beta$	0	μ	1	α	$\bar{\alpha}$
0	0	0	0	0	0	μ	1	0	1
μ	0	μ	μ	μ	μ	μ	1	μ	μ
1	0	μ	1	1	1	1	1	1	0

А значення формул, побудованих за алфавітом зв'язок $\{\Rightarrow,\Leftrightarrow\}$ над об'єктами висловлювань α і β, задаються табл. 1.2.

Таблиця 1.2

Значення формул висловлювань за зв'язками $\{\Rightarrow,\Leftrightarrow\}$

$\alpha\Rightarrow\beta$	0	μ	1	$\alpha\Leftrightarrow\beta$	0	μ	1
0	1	1	1	0	1	0	0
μ	0	1	1	μ	0	1	0
1	0	0	1	1	0	0	1

Дамо пояснення формул наведених в табл. 1.1 і 1.2.

Формула (*заперечення*) $\bar{\alpha}=\neg\alpha$ читається «не α» або «заперечення α» і тому формула має значення протилежне до значення $V(\alpha)$.

Формула (*кон'юнкції*) $\alpha\wedge\beta$ (інколи для скорочення запису застосовується позначення $\alpha\beta$) читається як «α і β» і приймає значення істини у тому і тільки тому випадкові, коли об'єкти α і β істині. Для знаходження значення цієї формули також можливо скористатись правилом $V(\alpha\wedge\beta)=\min\{V(\alpha),V(\beta)\}$.

Формула $\alpha\vee\beta$ (*диз'юнкції*) читається – «α або β» і приймає значення істини, коли хоча б один з об'єктів має значення істини, це правило також можливо записати у вигляді формули $V(\alpha\vee\beta)=\max\{V(\alpha),V(\beta)\}$.

Формула (*імплікації*) $\alpha\Rightarrow\beta$ може читатися – «якщо α, тоді β» або «з α слідує β» і має значення істини завжди, окрім випадків ($V(\alpha)=\mu$ або $V(\alpha)=1$ і $V(\beta)=0$) і ($V(\alpha)=1$ і $V(\beta)=\mu$), при яких ця формула має хибне значення.

І остання формула (*еквівалентності*) $\alpha\Leftrightarrow\beta$ може читатися як «α еквівалентне β», тому значення за цією формулою будуть істині завжди де $V(\alpha)=V(\beta)$ і хибними у всіх інших випадках.

Зрозуміло, що формули які виводяться в логічній формальній системі $\mathbb{S}_A$ є ланцюжками логічної формальної мови $\mathbb{L}$. Питання пов'язані з дослідженням зв'язок та результатів формул розглядаються в алгебраїчній теорії логіки [5].

1.4.3. Логічні функції висловлювань

Формули логіки висловлювань можливо пов'язати з *логічними функціями висловлювань* $f_j(\alpha_1, \alpha_2, \ldots, \alpha_n)$, де α_i – атомарні висловлювання. Наприклад, формулі $(\alpha \wedge \beta) \vee \gamma$ відповідає функція $f(\alpha, \beta, \gamma) = (\alpha \wedge \beta) \vee \gamma$, тобто функція f реалізує формулу-ланцюжок $(\alpha \wedge \beta) \vee \gamma$. Складне висловлювання розглядається як функція n – змінних x_i, тобто маємо $n-$ *місцину* функцію висловлювань f^n. Так постійне логічне значення $\alpha = 1$ (*логічна константа*) є 0– місциною логічною функцією, змінне атомарне висловлювання x або $\neg x$ – одномісною функцією, $x_1 \wedge x_2$, $x_1 \vee x_2$, $x_1 \Rightarrow x_2$ – двомісні функції і так далі. Область визначення логічної функції f^n – $Dom\, f$ залежить від розміру алфавіту, на якому визначені змінні x_i. Так на тернарному алфавітові T область $Dom\, f$ складається з 3^n комбінацій значень змінних, а область значень функції $Ran\, f$ може залежати від складових алфавіту зв'язок Z. Тому, як правило, при невеликому значені величини n, логічну функцію f^n задають у формі таблиці

$Dom\, f$	$Ran\, f$
$x_1 x_2 \ldots x_n$	$f(x_1, x_2, \ldots, x_n)$

, яка зветься *таблицею істинності*.

Приклад 1.2. Розглянемо систему масового обслуговування (СМО), яка обслуговує деякий потік заявок. СМО призначені для задоволення потоку заявок за допомогою одного або декількох каналів обслуговування (зв'язку), котрі виконують стандартні операції. СМО зустрічаються практично в усіх видах сучасної діяльності людини і набули широкого застосування в технічних, біологічних, економічних та інших галузях. Нехай задана модель СМО складається з трьох терміналів для прийому потоку заявок і каналу зв'язку, який відчиняється і пропускає заявки або запирається. Канал зв'язку відчиняється для пропуску групи заявок при наявності на терміналах неменше

двох заявок. Необхідно побудувати модель функціонування СМО у вигляді логічної функції.

Позначимо наявність заявки на першому терміналі через x_1, на другому – x_2 і на третьому – x_3, тоді моделюючою функцією функціонування СМО буде $f(x_1, x_2, x_3)$. Зрозуміло, що x_i може прийняти значення 0 (заявка відсутня на i - терміналі) або 1 (заявка присутня на i - терміналі) і значенню функції f буде також відповідати 0 (закритий канал) або 1 (відкритий канал). Тепер таблиця істинності функції f має вигляд зображений у табл. 1.3.

Таблиця 1.3

Таблиця істинності системи СМО

x_1	x_2	x_3	*Ran f*	x_1	x_2	x_3	*Ran f*
0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	1
0	1	0	0	1	1	0	1
0	1	1	1	1	1	1	1

Дві функції f_1^n і f_2^n вважаються однаковими ($f_1^n(\cdot) = f_2^n(\cdot)$), якщо $Dom\, f_1 = Dom\, f_2$ і при будь якому наборі $(\alpha_1, \alpha_2, \ldots, \alpha_n)$ значень змінних $x_1, x_2, \ldots, x_n$ з області їх визначення $V\left(f_1(\alpha_1, \alpha_2, \ldots, \alpha_3)\right) = V\left(f_2(\alpha_1, \alpha_2, \ldots, \alpha_3)\right)$. Для прикладу, функції $f_1(x_1, x_2) = x_1 \wedge x_2$ і $f_2(x_1, x_2) = (x_1 \wedge x_2) \vee ((x_1 \wedge x_2) \wedge (x_1 \wedge x_2))$ однакові.

▷ Дійсно кон'юнкція змінних x_1, x_2 при відповідному наборі (α_1, α_2) на числовому алфавітові T за табл. 1.1 приймає одне із значень цього алфавіту $\beta = V(\alpha_1 \wedge \alpha_2)$. Тому значення кон'юнкції $V((\alpha_1 \wedge \alpha_2) \wedge (\alpha_1 \wedge \alpha_2)) = \beta \wedge \beta = \beta = V(\alpha_1 \wedge \alpha_2)$, і далі значення диз'юнкції $V((\alpha_1 \wedge \alpha_2) \vee (\alpha_1 \wedge \alpha_2)) = \beta \vee \beta$ за тією ж таблицею дорівнює β. Отже $V\left(f_1(\alpha_1, \alpha_2)\right) = V\left(f_2(\alpha_1, \alpha_2)\right)$. ◁

Якщо кількість змінних функції зростає, зрозуміло, що розмір таблиці істинності стає великим і користуватися нею незручно. Більш зручним у цьому випадку є аналітичний спосіб представлення логічної функції. Якщо змінні x_i, наприклад, приймають значення $V(x_i)$ 0 або 1 для всіх $i = \overline{1, n}$, і функція f, яка побудована на зв'язках алфавіту $Z = \{\wedge, \vee, \neg\}$, приймає значення 0 або 1. Для цього випадку розроблені канонічні форми представлення логічних функцій, які називають

булєвими функціями в досконалій де'зюнктивній нормальній формі або скорочено *ДДНФ*, такі форми функції f^n створюються наступним чином. На наборі змінних $x_i;\ i=\overline{1,n}$ будується *елементарні кон'юнкції* $k_m=\bigwedge\limits_{j=1}^{m} x_{i_j}^*$ такі, що у виразі k_m кожен символ $x_{i_j}^*=\begin{cases}x_{i_j}\\ \neg x_{i_j}\end{cases};\ j=\overline{1,m}$ може зустрічатися тільки один раз. До елементарних кон'юнкцій відносяться: константа 1, одноелементні конструкції k_1, які складаються зі змінних x_i або їх заперечень $\overline{x}_i$ $(\neg x_i)$. Зрозуміло, що кількість різних елементарних кон'юнкцій дорівнює 2^n. Наприклад, за таблицею істинності розглянутого прикладу 1.2 набору значень (0 0 0) змінних x_1, x_2, x_3 відповідає елементарна кон'юнкція $k_1=\overline{x}_1\wedge\overline{x}_2\wedge\overline{x}_3$, набору (0 0 1) відповідає $k_2=\overline{x}_1\wedge\overline{x}_2\wedge x_3$ і так далі, останньому набору (1 1 1) табл. 1.3 відповідає $k_8=x_1\wedge x_2\wedge x_3$. З табл. 1.3 також видно, що з восьми елементарних кон'юнкцій тільки для кон'юнкцій k_4, k_6, k_7 і k_8 відчиняється канал зв'язку СМО.

Диз'юнкція елементарних кон'юнкцій $\bigvee\limits_{m=1}^{s} k_m$ зветься ДДНФ функції f^n, якщо значення k_m дорівнюють 1. Для розглянутої вправи ДДНФ логічної булєвої функції f^3 за табл. 1.3 має вигляд

$$f_1(x_1,x_2,x_3)=k_4\vee k_6\vee k_7\vee k_8=\overline{x}_1\wedge x_2\wedge x_3\vee x_1\wedge\overline{x}_2\wedge x_3\vee \\ \vee x_1\wedge x_2\wedge\overline{x}_3\vee x_1\wedge x_2\wedge x_3. \tag{1.3}$$

Тут враховано те, що кон'юнкція (як операція множення) старша за диз'юнкцію, тому у виразі (1.3) дужки не застосовуються.

Якщо ж у визначені ДДНФ замінити кон'юнкції на диз'юнкції і навпаки і відповідно нулі одиницями та навпаки, то будемо мати *досконалу кон'юнктивну нормальну форму* (ДКНФ) функції f^n. Так ДКНФ логічної функції прикладу 1.2 подвійна до функції (1.3) є

$$\overline{f}_1(x_1,x_2,x_3)=\overline{k}_2\wedge\overline{k}_5\wedge\overline{k}_3\wedge\overline{k}_1= \\ =(\overline{x}_1\vee\overline{x}_2\vee x_3)\wedge(x_1\vee\overline{x}_2\vee\overline{x}_3)\wedge(\overline{x}_1\vee x_2\vee\overline{x}_3)\wedge(\overline{x}_1\vee\overline{x}_2\vee\overline{x}_3) \tag{1.4}$$

Функції (1.3) і (1.4) моделюють процес відкриття і запирання каналу зв'язку в системі СМО, тому аналітичний вираз логічної функції можливо представити як

$$f(x_1, x_2, x_3) = \begin{cases} f_1(x_1, x_2, x_3) = 1, \textit{ канал відчинений}; \\ \overline{f}_1(x_1, x_2, x_3) = 0, \textit{ канал зачинений}. \end{cases}$$

Раніше для часткового випадку, показано що функція логіки висловлювань може бути представлена у ДДНФ. Виникає питання чи завжди це можливо. Відповідь на яке дає лема про розкладення логічної функції [9].

Лема 1.1. Довільна логічна функція f^n може бути представлена через елементарні кон'юнкції $k_m^{\alpha} = \bigwedge_{j=1}^{m} x_{i_j}^*$, $x_{i_j}^* = \begin{cases} \alpha_{i_j} \\ \neg\alpha_{i_j} \end{cases}$; $j = \overline{1,m}$, де $\alpha_{i_j} = 0, 1$, у вигляді

$$f(x_1, x_2, \ldots, x_k, x_{k+1}, \ldots, x_n) = \bigvee_{m=1}^{s} (k_m^{\alpha} \wedge f(\alpha_1, \alpha_2, \ldots, \alpha_k, x_{k+1}, \ldots, x_n)) \qquad (1.5)$$

▷ Задамо довільний набір $(\beta_1, \beta_2, \ldots, \beta_k)$ на алфавіті {0,1} такий, що $x_1 = \beta_1, \ldots, x_k = \beta_k$; тоді в лівій частині виразу (1.5) маємо $f(\beta_1, \beta_2, \ldots, \beta_k, x_{k+1}, \ldots, x_n)$, а в правій частині набори кон'юнкцій $\bigwedge_{j=1}^{m} \beta_{i_j}^* \wedge f(\beta_1, \beta_2, \ldots, \beta_k, x_{k+1}, \ldots, x_n)$ і нехай для визначеності $\alpha_i = 1$, при цьому можливі дві ситуації. У першій ситуації $\beta_{i_j}^* = \alpha_{i_j}^* = \alpha_i = 1$, тоді $\beta_{i_j}^* = \alpha_{i_j}^* = \alpha_i = 1$ і тому елементарні кон'юнкції дорівнюють 1; отже ліва і права частини виразу (1.5) однакові. В другій ситуації, хоча б одне з значень $\beta_i \neq \alpha_i$, тому диз'юнкція елементарних кон'юнкцій дає (див. табл. 1.1)

$$0 \vee f(\beta_1, \beta_2, \ldots, \beta_k, x_{k+1}, \ldots, x_n) = f(\beta_1, \beta_2, \ldots, \beta_k, x_{k+1}, \ldots, x_n).$$

Таким чином вираз (1.5) має місце при будь яких обставинах. ◁

Теорема 1.1. Будь яка булєва логічна функція $f^n(\cdot) \not\equiv 0$ може бути однозначно представлена у ДДНФ.

Доведення легко отримати спираючись на лему 1.1, за якою при $k = n$ з виразу (1.5) маємо $f(x_1, x_2, \ldots, x_n) = \bigvee_{m=1}^{s} k_m^x$.

1.4.4. Завдання і вправи

1. На трійковому алфавітові $T = \{0, \mu, 1\}$ визначити значення висловлювань:
   а) сніг – білий, чорний, брудний;
   б) трава – зелена, біла, жовта, рожева;
   в) небо – блакитне, чорне, сіре, кольорове.
2. Записати у вигляді формул логіки висловлювання:
   а) погода сонячна і тепла;
   б) якщо студент старанний, тоді він вчиться добре;
   в) студент відмінник, тоді і тільки тоді коли вчиться на «відмінно»;
   г) студент вчиться добре, тоді він отримує оцінки «добре» або оцінки «добре» і «відмінно»;
   д) якщо чотирикутник паралелограм, тоді він може бути ромбом або ромбом і квадратом або прямокутником.
3. На тринарному алфавітові $T = \{0, \mu, 1\}$ визначити значення формул вправи 2.
4. На сукупності чотирикутників побудувати формальну систему логіки висловлювань з врахуванням п'ятого висловлювання вправи 2.
5. На сукупності студентів групи побудувати формальну систему логіки висловлювань з врахуванням висловлювань 2), 3) і 4) вправи 2.
6. Записати таблиці істинності для наступних логічних формул:

   1) $((\alpha \Rightarrow \beta) \Leftrightarrow (\bar{\alpha} \vee \beta))$;

   2) $((\alpha \Rightarrow \beta) \vee \bar{\alpha})$;

   3) $(\alpha \Rightarrow (\beta \wedge \gamma))$;

   4) $(\bar{\alpha} \Rightarrow (\beta \vee \gamma))$;

   5) $((\alpha \wedge \beta) \Leftrightarrow (\beta \wedge \gamma))$;

   6) $((\alpha \wedge \beta) \Leftrightarrow (\gamma \Rightarrow (\bar{\alpha} \vee \beta)))$;

   7) $((\alpha \Rightarrow \beta) \vee (\alpha \Rightarrow (\alpha \wedge \beta)))$;

   8) $((\bar{\beta} \Rightarrow \alpha) \vee \beta) \wedge (\alpha \wedge (\beta \vee \bar{\alpha})))$;

   9) $(((\alpha \wedge \bar{\beta}) \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \beta))$.

7. На тернарному алфавіті T записати таблиці істинності для логічних функцій:

   а) $x_1x_2x_3x_4 \Leftrightarrow x_5$;

   б) $(x \Rightarrow \neg y) \vee (xyz) \wedge (x \vee y \vee \overline{z})$;

   в) $(((x \wedge yx) \Rightarrow z) \Rightarrow (x \Rightarrow yz))$;

   г) $(((x_1x_2) \Rightarrow x_3) \vee y) \vee (y \wedge ((x_1x_2) \vee \overline{x}_3)))$;

   д) $(((z \wedge y) \Rightarrow (xy)) \Leftrightarrow (\overline{xy\overline{z}} \vee y \vee z))$;

   е) $\overline{x_1\overline{x}_2x_3\overline{x}_4} \Leftrightarrow (\neg x_5 \vee x_4)$;

   ж) $((xzy \Rightarrow u) \vee ((x \vee (yz)) \Rightarrow (\neg x \wedge (\overline{xyz}))))$.

8. Для формул вправи 6) записати їх логічні функції реалізації.
9. На числовому алфавіті $\{0,1\}$ задати логічні функції вправи 8 у вигляді таблиць істинності.
10. Побудувати ДДНФ і ДКНФ булєвих функцій f^3, визначених таблицями істинності:

1)

x_1	x_2	x_3	$Ran\ f$	x_1	x_2	x_3	$Ran\ f$
0	0	0	1	1	0	0	0
0	0	1	0	1	0	1	1
0	1	0	1	1	1	0	0
0	1	1	0	1	1	1	1

;

2)

x_1	x_2	x_3	$Ran\ f$	x_1	x_2	x_3	$Ran\ f$
0	0	0	0	1	0	0	0
0	0	1	1	1	0	1	1
0	1	0	0	1	1	0	0
0	1	1	1	1	1	1	1

;

3)

x_1	x_2	x_3	$Ran\ f$	x_1	x_2	x_3	$Ran\ f$
0	0	0	1	1	0	0	1
0	0	1	1	1	0	1	0
0	1	0	1	1	1	0	0
0	1	1	0	1	1	1	0

.

1.5. Рівносильні формули висловлювань

Введена формальна система логіки дозволяє будувати формули логічних висловлювань, над якими можливо виконувати алгебраїчні перетворення. Зрозуміло, що ці перетворення не передбачені у наведеній формальній системі, тобто вони знаходяться зовні системи, щоб показати можливість зв'язку формальної системи з алгеброю виконаємо перетворення на рівносильних формулах висловлювань.

1.5.1. Закони рівносильності

Нехай через α і β позначено дві формули логіки висловлювань. Тоді маємо.

Визначення 1.6. Формули α і β звуться *рівносильними*, якщо $V(\alpha) = V(\beta)$ при будь яких значеннях елементарних об'єктів, з яких складаються ці формули.

Визначена рівносильність встановлює з в'язок між формулами α і β з одного боку, а з іншого – задає однаковість для відповідних реалізуючих логічних функцій. Цей зв'язок позначимо знаком (=), таким чином маємо $\alpha = \beta$. Рівносильність формул задовольняє наступним умовам [5, 6]:

а) $\alpha = \alpha$ *(рефлективність)*;

б) $(\alpha = \beta) \Rightarrow (\beta = \alpha)$ *(симетричність)*;

в) $(\alpha = \beta \wedge \beta = \gamma) \Rightarrow (\alpha = \gamma)$ *(транзитивність)*.

Наведемо тепер важливі приклади законів, яким задовольняють рівносильні формули:

а) $\alpha = \overline{\overline{\alpha}}$ (*закон подвійного заперечення*);

б) $\alpha \wedge \beta = \beta \wedge \alpha$,
$\alpha \vee \beta = \beta \vee \alpha$ (*закон комутативності*);

в) $(\alpha \wedge \beta) \wedge \gamma = \alpha \wedge (\beta \wedge \gamma)$,
$(\alpha \vee \beta) \vee \gamma = \alpha \vee (\beta \vee \gamma)$ (*закон асоціативності*); (1.6)

г) $\alpha \wedge (\beta \vee \gamma) = (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$,
$\alpha \vee (\beta \wedge \gamma) = (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$ (*закон дистрибутивності*);

д) $\overline{\alpha \wedge \beta} = \bar{\alpha} \vee \bar{\beta}$,
$\overline{\alpha \vee \beta} = \bar{\alpha} \wedge \bar{\beta}$ (*закон де Моргана*);

е) $\alpha \wedge \alpha = \alpha$,
$\alpha \vee \alpha = \alpha$ (*закон ідемпотентності*);

ж) $\alpha \Rightarrow \beta = \bar{\beta} \Rightarrow \bar{\alpha}$ (*закон обертання*);

з) $\alpha \Rightarrow \beta = \bar{\alpha} \vee \beta$;

и) $\alpha \Leftrightarrow \beta = (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$

к) $\alpha \wedge \bar{\alpha} = 0$, $\alpha \vee \bar{\alpha} = 1$; $V(\alpha) \neq \mu$ (*закон виключення третього*).

1.5.2. Логічні перетворення

В межах формальних систем можливо не тільки виводити логічні формули, але й виконувати перетворення формул спрощуючи їх. Наприклад, булєва функція (1.3) виводиться в системі $\mathbb{S}_A$ та може бути спрощена після застосування законів е) і а) – г)

$$f(x_1, x_2, x_3) = \bar{x}_1 x_2 x_3 \vee x_1 \bar{x}_2 x_3 \vee x_1 x_2 \bar{x}_3 \vee x_1 x_2 x_3 =$$
$$= \bar{x}_1 x_2 x_3 \vee x_1 \bar{x}_2 x_3 \vee x_1 x_2 \bar{x}_3 \vee x_1 x_2 x_3 \vee x_1 x_2 x_3 \vee x_1 x_2 x_3 =$$
$$= (x_1 \vee \bar{x}_1) x_2 x_3 \vee (x_2 \vee \bar{x}_2) x_1 x_3 \vee (x_3 \vee \bar{x}_3) x_2 x_1 = x_2 x_3 \vee x_1 x_3 \vee x_2 x_1 =$$
$$= x_2 x_3 \vee (x_3 \vee x_2) x_1 .$$

В процесі перетворення формул інколи зручно скористатися правилом підстановки формул замість змінного об'єкту та правилом заміни підформул (часток формули) .

Твердження 1.1. Правило підстановки формули замість змінного об'єкту. Якщо α і β – рівносильні формули, в які входить змінний об'єкт α_1, тоді в результаті одночасної підстановки формули γ замість об'єкту α_1 у формули α і β отримаємо рівносильні формули.

Наприклад, якщо підставити у рівносильну формулу $\alpha \vee \beta = \beta \vee \alpha$ замість об'єкту α формулу $\delta \Rightarrow \gamma$, тоді отримаємо наступний рівносильний вираз логічної формули: $(\delta \Rightarrow \gamma) \vee \beta = \beta \vee (\delta \Rightarrow \gamma)$.

Твердження 1.2. Правило заміни підформул. Нехай α є підформулою формули γ_α і формула α замінена на рівносильну формулу β. В результаті отримаємо формулу γ_β, рівносильну до формули γ_α.

Пояснимо застосування цього твердження на прикладі формули $\overline{\alpha \wedge \beta} \Rightarrow \gamma$, до якої застосуємо рівносильну заміну $\overline{\alpha \wedge \beta} = \bar{\alpha} \vee \bar{\beta}$ і після заміни отримаємо рівносильну формулу $(\overline{\alpha \wedge \beta}) \Rightarrow \gamma = (\bar{\alpha} \vee \bar{\beta}) \Rightarrow \gamma$.

Наведемо приклад спрощення формули за цими правилами та законами рівносильності:

$$\overline{\bar{\alpha} \wedge \bar{\beta}} = \bar{\bar{\alpha}} \vee \bar{\bar{\beta}} = \bar{\bar{\alpha}} \vee \beta = \alpha \vee \beta .$$

Твердження 1.1 і 1.2 для логічних функцій f^n, $f_1^{m_1}$, $f_2^{m_2}$, …, $f_n^{m_n}$, відповідає діям:

1) підстановки у вираз однакових функції $f(x_1, x_2, \ldots, x_n)$ замість змінних x_i функцій $f_j(x_1, x_2, \ldots, x_{m_j})$;

2) підстановки у вираз функції $f(x_1, x_2, \ldots, x_n)$ замість яких то формул над змінними $x_1, x_2, \ldots, x_{m_j}$ однакових функцій $f_j(x_1, x_2, \ldots, x_{m_j})$.

1.5.3. Завдання і вправи

1. Пересвідчитись у тому, що рівносильність формул висловлювань задовольняють умовам: рефлективності, симетрії та транзитивності.
2. Довести закони рівносильності 1) – 9).
3. Довести, що $\alpha = \beta$ тоді і тільки тоді, коли $V(\alpha \Leftrightarrow \beta) = 1$ для будь яких значеннях 0 або 1 висловлювань α і β.
4. Довести, що:

   а) $\neg((\alpha \wedge \beta) \vee \gamma) = (\bar{\alpha} \vee \bar{\beta}) \wedge \bar{\gamma}$;

   б) $(\bar{\alpha} \vee \beta) \wedge (\bar{\beta} \vee \alpha) = \alpha \Leftrightarrow \beta$;

   в) $\neg(\neg\bar{\alpha} \wedge \alpha) \vee (\beta \vee \beta) = (\alpha \Rightarrow \beta)$.

5. Довести еквівалентності формул:

   а) $(\alpha \Rightarrow \bar{\alpha}) = \bar{\alpha}$;

б) $(\alpha \vee (\beta \wedge \overline{\beta})) = \alpha$;

в) $(\alpha \wedge (\beta \vee \overline{\beta})) = \alpha$;

г) $(\alpha \vee (\overline{\alpha} \wedge \beta)) = (\alpha \vee \beta)$;

д) $((\alpha \vee \beta) \wedge (\alpha \vee \overline{\beta})) = \alpha$.

6. Довести однаковість функцій:

   а) $f_1(x_1, x_2) = x_1 \vee (\overline{x}_1 \wedge x_2)$ і $f_2(x_1, x_2) = x_1 \vee x_2$;

   б) $f_1(x_1, x_2) = (\overline{x}_1 \vee x_2) \wedge (\overline{x}_2 \vee x_1)$ і $f_2(x_1, x_2) = x_1 \Leftrightarrow x_2$;

   в) $f_1(x_1, x_2) = x_1 \Rightarrow x_2$ і $f_2(x_1, x_2) = \neg(\neg\overline{x}_1 \wedge x_1) \vee (x_2 \vee x_2)$.

7. Зв'язка $(\bullet)$ між логічними змінними x_1, x_2, x_3 у вигляді $x_1 \wedge x_2 \vee x_1 \wedge x_3 \vee x_2 \wedge x_3$ називається мажоритарною і позначається як $x_1 \bullet x_2 \bullet x_3$. Встановити рівносильність наступних формул:

   а) $x_1 \bullet x_1 \bullet x_3 = x_1$;

   б) $x_1 \bullet \overline{x}_1 \bullet x_3 = x_3$;

   в) $\neg(x_1 \bullet x_2 \bullet x_3) = \overline{x}_1 \bullet \overline{x}_2 \bullet \overline{x}_3$.

8. Записати закони рівносильності для арифметичних операцій: $(+)$ – додавання, $(-)$ – віднімання та $(\cdot)$ – добутку при їх логічних інтерпретаціях:

$$x \cdot y = x \wedge y;\ 1 - x = \neg x;\ x + y - x \cdot y = x \vee y.$$

1.6. Логічні числення

При створені формальної системи, найбільш складно відтворити систему аксіом. Розглянемо питанням побудови системи аксіом формальної логічної системи. Для цього скористуємося підходом, який використовується в численні висловлювань.

1.6.1. Числення висловлювань

Численням висловлювань є формальна система, в якій об'єктами виступають формальні формули (терми), правила їх застосування і аксіоми.

Визначення 1.7. Виводом формули τ в системі $\mathbb{S}$ зветься така послідовність формул $\tau_1, \tau_2, \ldots, \tau_n$, що кожна з них отримана або з аксіоми формальної системи, або з попередніх раніше утворених формул цієї послідовності і записується як $\tau_1, \tau_2, \ldots, \tau_n \to \tau$.

Може статися, що не існує процедури виводу наперед заданої формули, тоді кажуть, що проблема виводу нерозв'язана. Формула τ, для якої проблема виводу розв'язана – зветься *теоремою* формальної системи $\mathbb{S}$. В залежності від змісту сукупності зв'язок можуть розглядатися різні системи числення висловлювань.

Нехай, наприклад, задано алфавіт зв'язок $Z = \{\wedge, \vee, \neg, \Rightarrow\}$, алфавіт змінних логічних об'єктів $A = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ і додатковий алфавіт $B = \{(,), \to\}$, тоді числення висловлювань над алфавітами A, Z, B можливо визначити наступною конструктивною системою аксіом:

$$
\begin{aligned}
&1)\ (\alpha \wedge \beta) \Rightarrow \alpha,\\
&2)\ (\alpha \wedge \beta) \Rightarrow \beta,\\
&3)\ \alpha \Rightarrow (\alpha \vee \beta),\\
&4)\ \beta \Rightarrow (\alpha \vee \beta),\\
&5)\ (\neg\neg \alpha) \Rightarrow \alpha,\\
&6)\ \alpha \Rightarrow (\beta \Rightarrow \alpha),\\
&7)\ (\alpha \Rightarrow \neg\beta) \Rightarrow (\beta \Rightarrow \neg\alpha),\\
&8)\ (\alpha \Rightarrow \beta) \Rightarrow ((\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow (\alpha \Rightarrow \gamma)),\\
&9)\ (\alpha \Rightarrow \beta) \Rightarrow ((\alpha \Rightarrow \gamma) \Rightarrow (\alpha \Rightarrow (\beta \wedge \gamma))),\\
&10)\ (\alpha \Rightarrow \gamma) \Rightarrow ((\beta \Rightarrow \gamma) \Rightarrow ((\alpha \vee \beta) \Rightarrow \gamma)),
\end{aligned}
\qquad (1.7)
$$

в яких через α, β, γ позначено довільні формули, і формула виводу β є безпосереднім наслідком формул α і $(\alpha \Rightarrow \beta)$, що можливо записати у вигляді схеми Поста:

$$\frac{\alpha,\ (\alpha \Rightarrow \beta)}{\beta} \tag{1.8}$$

Наприклад, якщо висловлювання δ і $\delta \Rightarrow (\gamma \Rightarrow \delta)$ істинні, то висловлювання $\gamma \Rightarrow \delta$ також вважається істинним.

У формальній системі числення початковою аксіомою може бути будь яка формула, отримана за наведеною схемою аксіом 1) – 10) з формул (1.7) після підстановки в неї конкретних формул, побудованих на алфавітах A, B і Z. Так, аксіомами є формули:

$$(\tau_1 \wedge \tau_2) \Rightarrow ((\tau_3 \Rightarrow \tau_4) \vee (\tau_1 \wedge \tau_2)),\ (\tau_1 \vee \tau_2) \Rightarrow ((\tau_3 \wedge \tau_4) \Rightarrow (\tau_1 \vee \tau_2))$$

отримані за схемами аксіом 4) і 6) відповідно.

Виводом формули, як було раніше вказано, в численні висловлювань є скінчена послідовність формул $\tau_1, \tau_2, \ldots, \tau_n$ таких, що для кожного індексу i $(1 \le i \le n)$ τ_i є: або аксіомою, або безпосереднім виводом із формул τ_j, τ_k за правилом (1.7) і формула τ_n співпадає з формулою τ.

Приклад 1.3. Показати, що формула $(\alpha \Rightarrow \alpha) = \tau$ виводиться у заданому вище численні висловлювань.

▹ Для цього спочатку скористуємося схемами аксіом 6) та 8). За аксіомою 6) маємо вивід: $\rightarrow (\alpha \Rightarrow (\alpha \Rightarrow \alpha))$, приймемо формулу $(\alpha \Rightarrow (\alpha \Rightarrow \alpha))$ за початок виводу τ_1 і за схемою 8), після застосування підстановок $\beta = (\alpha \Rightarrow \alpha)$ і $\gamma = \alpha$, отримаємо аксіому:

$$\rightarrow (\alpha \Rightarrow (\alpha \Rightarrow \alpha)) \Rightarrow ((\alpha \Rightarrow ((\alpha \Rightarrow \alpha) \Rightarrow \alpha)) \Rightarrow (\alpha \Rightarrow \alpha)),$$

з якої видно, що формула

$$\tau_2 = (\tau_1 \Rightarrow ((\alpha \Rightarrow ((\alpha \Rightarrow \alpha) \Rightarrow \alpha)) \Rightarrow (\alpha \Rightarrow \alpha)))$$

виводиться у численні висловлювань. Скористуємося тепер правилом виводу (1.8), за яким з послідовності формул τ_1, τ_2 отримаємо вивід $\tau_1, \tau_2 \rightarrow ((\alpha \Rightarrow ((\alpha \Rightarrow \alpha) \Rightarrow \alpha)) \Rightarrow (\alpha \Rightarrow \alpha))$, тобто маємо, що формула $\tau_3 = ((\alpha \Rightarrow ((\alpha \Rightarrow \alpha) \Rightarrow \alpha)) \Rightarrow (\alpha \Rightarrow \alpha))$ виводиться у даному численні. Зрозуміло, що формула $(\alpha \Rightarrow ((\alpha \Rightarrow \alpha) \Rightarrow \alpha)) = \tau_4$ також виводиться за аксіомою 6) при значені $\beta = (\alpha \Rightarrow \alpha)$ отримаємо вивід $\rightarrow (\alpha \Rightarrow ((\alpha \Rightarrow \alpha) \Rightarrow \alpha))$. І знову скористуємося правилом (1.8), але

застосуємо його до послідовності τ_4, τ_3, тоді одержимо $\tau_4, \tau_3 \to (\alpha \Rightarrow \alpha)$. Отже формула $\tau_5 = (\alpha \Rightarrow \alpha)$ виводиться у численні висловлювань за допомогою послідовності виводу $\tau_1, \tau_2, \tau_3, \tau_4, \tau_5$. ◁

Для коректного визначення введеної у пункті 1.4.1 на алфавітові зв'язок Z формальної системи $\mathbb{S}_A$, необхідно включити до системи аксіом Λ систему формул (1.7) і правило виводу (1.8).

1.6.2. Деякі логічні системи та зв'язок між ними

Можливо побудувати інші логічні формальні системи. Наприклад, якщо за алфавіт зв'язок вибрати сукупність $\{\wedge, \vee, \neg\}$ $\{\wedge, \vee, \neg\}$ і в систему аксіом включити формули

$$
\begin{aligned}
&1)\ \alpha \wedge \alpha \Rightarrow \alpha,\ \alpha \vee \alpha \Rightarrow \alpha;\\
&2)\ \alpha \wedge \beta \Rightarrow \beta \wedge \alpha,\ \alpha \vee \beta \Rightarrow \beta \vee \alpha;\\
&3)\ \alpha \Rightarrow \alpha \wedge \beta,\ \alpha \Rightarrow \alpha \vee \beta;\\
&4)\ (\beta \Rightarrow \gamma) \Rightarrow (\alpha \wedge \beta \Rightarrow \alpha \wedge \gamma),\ (\beta \Rightarrow \gamma) \Rightarrow (\alpha \vee \beta \Rightarrow \alpha \vee \gamma);\\
&5)\ \alpha \wedge (\alpha \vee \beta) \Rightarrow \alpha,\ \alpha \vee (\alpha \wedge \beta) \Rightarrow \alpha;\\
&6)\ (\neg\neg\alpha) \Rightarrow \alpha,
\end{aligned}
\tag{1.9}
$$

де під імплікацією $\alpha \Rightarrow \beta$ розуміється еквівалентна формула $\neg\alpha \vee \beta$, тоді отримаємо класичну *булєву формальну систему* $\mathbb{S}_A^b$.

Якщо на двійковому алфавітові $\{0,1\}$ ввести в розгляд алфавіт Жигалкіна $Z_1 = \{\oplus, \wedge, 1\}$, де $(\oplus)$ – *зв'язка складання за* $\mathrm{mod}\,2$ такий, що мають місце наступні рівносильні перетворення над елементами двійкового алфавіту

$$
\begin{cases}
0 \oplus 0 = 0,\ 1 \oplus 0 = 1,\\
0 \oplus 1 = 1,\ 1 \oplus 1 = 0.
\end{cases}
\tag{1.10}
$$

Якщо в систему аксіом Λ включити формули 1) – 4) з виразів (1.9) відносно зв'язки $(\wedge)$, а також врахувати рівносильні формули відносно зв'язки $(\oplus)$:

$$
1)\ \alpha \oplus \beta = \beta \oplus \alpha; \tag{1.11}
$$

2) $\alpha \oplus (\beta \oplus \gamma) = (\alpha \oplus \beta) \oplus \gamma$;

3) $\alpha \wedge (\beta \oplus \gamma) = \alpha \wedge \beta \oplus \alpha \wedge \gamma$;

4) $\alpha \oplus \alpha = 0$;

5) $\alpha \oplus 0 = \alpha$,

тоді отримаємо формальну систему Жигалкіна $\mathbb{S}_A^g$. Логічна функція виведена в системі $\mathbb{S}_A^g$ зветься поліномом Жигалкіна. Формальна система Жигалкіна зручно застосовувати в системах програмування [1].

Існує певний зв'язок між формальними системами $\mathbb{S}_A^b$ і $\mathbb{S}_A^g$. Для того, щоб з'ясувати цей зв'язок введемо одне визначення.

Визначення 1.8. Формальні системи $\mathbb{S}_A = \langle A, A^*, \Lambda, P \rangle$ і $\mathbb{S}_A^1 = \langle A, A_1^*, \Lambda_1, P_1 \rangle$ з відповідними мовами $A^* = \{A, Z\}^*$ і $A_1^* = \{A_1, Z_1\}^*$ еквівалентні, якщо існують такі взаємно однозначні перетворення r над їх алфавітами зв'язок, що будь який ланцюжок l формальної мови A^*, виведений в системі $\mathbb{S}_A$, перетворюється в ланцюжок $l_1 = r(l)$ мови A_1^* і навпаки.

Теорема 1.2. Логічні формальні системи $\mathbb{S}_A^b$ і $\mathbb{S}_A^g$ еквівалентні за визначенням 1.8 для будь якої формули α такої, що $V(\alpha) \not\equiv 0$.

▹ Нехай функція $f^n(\cdot)$ реалізує формулу α виведену в системі $\mathbb{S}_A^b$ тоді за теоремою 1.1 її можливо представити у ДДНФ, тобто $f(x_1, x_2, \cdots, x_n) = \bigvee_{m=1}^{s} k_m$. В цьому виразі використовується зв'язка $(\vee)$, а в кон'юнктивні набори k_m можливо входять атомарні елементи $\overline{x}_i$ і $V(k_m) = 1$, за перетворення r виберемо рівносильні перетворення $(=)$, яке за умовами 1) – 3) взаємно однозначне. Конкретно маємо перетворення

$$\neg\alpha = \alpha \oplus 1,\ \alpha \vee \beta = \alpha \wedge \beta \oplus \alpha \oplus \beta \qquad (1.12)$$

Очевидно, що перетворення (1.12) не впливають на правила виводу. Отже будь яка функція f^n може бути виведена в системі $\mathbb{S}_A^g$. Нескладно перевірити, що і навпаки поліному Жигалкіна за перетвореннями (1.12) можливо поставити у відповідність булєву функцію. ◃

Крім вище означених формальних систем, можливо ввести системи: $\mathbb{S}_A^h$ – Шеффера зі зв'язкою $(|)$, для якої мають місце перетворення

$\begin{cases} 0|0=1,\ 0|1=1, \\ 1|0=1,\ 1|1=0; \end{cases}$ і рівносильна формула $\alpha|\beta=\neg\alpha\vee\neg\beta$; $\mathbb{S}_A^p$ – Пірса-Вебба зі зв'язкою $(\downarrow)$ визначену $\begin{cases} 0\downarrow 0=1,\ 0\downarrow 1=0, \\ 1\downarrow 0=0,\ 1\downarrow 1=0; \end{cases}$ і формулами $\alpha\downarrow\beta=\neg(\alpha\vee\beta)=\neg\alpha\wedge\neg\beta$, і інші логічні формальні системи.

Для формальних систем важливим є питання повноти за алфавітами зв'язок. Алфавіт Z зветься *повним*, якщо ні одна з його зв'язок не може представлятися через рівносильне висловлювання над іншими зв'язками, і формальна система з цим алфавітом зв'язок зветься *повною по алфавіту* Z.

Так розглянута вище формальна система $\mathbb{S}_A$ є неповною тому, що алфавіт $Z=\{\wedge,\vee,\neg,\Rightarrow\}$ – неповний. Дійсно за рівносильними висловлюваннями маємо $\alpha\wedge\beta=\overline{\overline{\alpha}\vee\overline{\beta}}$ і $\alpha\vee\beta=\overline{\overline{\alpha}\wedge\overline{\beta}}$, тому за алфавіт Z можна взяти $\{\vee,\neg,\Rightarrow\}$ або $\{\wedge,\neg,\Rightarrow\}$ і відповідно перетворити систему аксіом 1) – 10) з виразів (1.7).

Існують критерії [9, 10], за допомогою яких вдається з'ясувати повноту формальної булєвої системи.

1.6.3. Завдання і вправи

1. Довести, що формули:

   а) $(\alpha\Rightarrow\alpha)$,

   б) $(\alpha\Rightarrow\neg\overline{\alpha})$,

   в) $((\alpha\vee\alpha)\Rightarrow\alpha)$,

   г) $(\alpha\vee\overline{\alpha})$,

   д) $(\neg(\alpha\wedge\overline{\alpha}))$,

   у яких α довільна формула, виводяться у формальній системі з аксіомами 1) – 10) і правилом виводу (1.8).
2. Показати, що наступні формули виводяться у формальній системі вправи 1:

   а) $((\alpha\vee\alpha)\sim\alpha)$,

   б) $((\alpha\wedge\alpha)\sim\alpha)$,

в) $((\alpha \vee \beta) \sim (\beta \vee \alpha))$,

г) $((\alpha \wedge \beta) \sim (\beta \wedge \alpha))$,

д) $(\neg\bar{\alpha} \sim \alpha)$,

е) $((\alpha \vee (\alpha \wedge \beta)) \sim \alpha)$,

ж) $((\alpha \wedge (\alpha \vee \beta)) \sim \alpha)$,

тут α і β – довільні формули.

3. Перетворити аксіоми 1) – 10) за допомогою зв'язок $\{\wedge, \neg, \Rightarrow\}$.
4. Записати аксіома 1) – 10) із застосуванням зв'язок $\{\vee, \neg, \Rightarrow\}$.
5. Показати, що формули вправи 1. є теоремами в формальній системі з аксіомами вправи 3.
6. Показати, що формули вправи 2. є теоремами в формальній системі з аксіомами вправи 3.
7. Показати, що формули вправи 1. є теоремами в формальній системі з аксіомами вправи 4.
8. Показати, що формули вправи 2. є теоремами в формальній системі з аксіомами вправи 4.
9. Задати формальну систему Шеффера.
10. Задати формальну систему Пірса – Вебба.

1.7. Елементи логіки предикатів

Сутність поняття предикату бере початок від еллінського вченого Аристотеля [2]. За предикат вибирається деяка $n-$ місцина функція, значеннями якої є висловлювання про значення $n-$ предметних об’єктів, зв’язок між якими задається у вигляді відношень або властивостей [3].

1.7.1. Загальні відомості і визначення, числення предикатів

Предикат пов'язаний з певною предметною областю, тобто областю на якій він визначений. У такій ролі може виступати будь яка формалізована область людських знань. Для позначення об’єктів предметних областей використовують уніфіковані позначки з симво-

лів деяких алфавітів A_i, так *предметні константи* позначаються символами $a_j \in A_i$, *предметні змінні* символи – $x_j \in A_i$, а для позначення новоутворень, операцій застосовують *функціональні символи* $f_i^k \in A_m$. Верхній індекс k вказує на місцину функціональної залежності f_i^k.

Подальшу уніфікацію позначень об'єктів предметних областей можливо виконати за допомогою *термів.* Терми будуються на введених вище предметних символах констант, змінних та функціональних символах за допомогою додатних символів: $(,)$ і за наступною формальною схемою правил:

1) терм τ_j є предметна константа $a_j \in A_i$;
2) терм τ_j є предметна змінна $x_j \in A_i$;
3) якщо $\tau_1, \tau_2, \ldots, \tau_n$ послідовність термів і f_j^n функціональний символ, тоді терм τ_j є вираз виду $f_j^n(\tau_1, \tau_2, \ldots, \tau_n)$.

Нехай кортеж $(x_1, x_2, \ldots, x_n) \in A_1 \times A_2 \times \ldots \times A_n = M$ – предметних змінних на алфавітах A_i і тернарний алфавіт $T = \{0, \mu, 1\}$, тоді

Визначення 1.9. предикатом $p^n(x_1, x_2, \ldots, x_n)$ місцини n на кортежі M зі значеннями з множини T зветься функція $p^n : M \to T$.

Предикат місцини n може бути визначений і на сукупності термів $(\tau_1, \tau_2, \ldots, \tau_n)$.

Так наступні формули задають предикати:

1) $p(x) = (|x| > 0)$ – одномісний або унарний;
2) $p(x, y) = ((x + y)^2 = x^2 + 2xy + y^2)$ – двохмісний або бінарний;
3) $p(\tau_1, \tau_2, \tau_3, \tau_4) = (\tau_1^2 + \tau_2^2 + \tau_3^2 + \tau_4^2 \geq 0)$ – чотирьохмісний;
4) $p(x_1, x_2, ..., x_k) =$ (якщо $0 \leq x_i \leq 1$, $f_1(x_1, x_2, ..., x_k) = \sum_{i=1}^{k} x_i = 1$, тоді предикат $f_2(x_1, x_2, ..., x_k) = -\sum_{i=1}^{k} x_i \log x_i \geq 0$) – місцини k.

Постійні предметні значення a_i, $i = 1, 2, \ldots, n$ також належать до сукупності M. У випадку, коли яка то змінна $x_k \in M$, $1 \leq k \leq n$ приймає постійне значення $a_k \in M$, тоді предикат місцини n стає $n-1$ - місцинним. Якщо змінні предикату приймають постійні значення, то предикат є нуль - місцинним.

Визначення 1.10. Підмножина $(\tau_1, \tau_2, \ldots, \tau_n) \in M_1 \subseteq M$, на яких предикат $p(\tau_1, \tau_2, \ldots, \tau_n)$ приймає значення 1 є *1–істинно* $(0-$*хибно*$, \mu-$*невизначено*$)$, зветься *множиною істинності* (*хибності, невизначеності*). У випадку співпадання множин, тобто $M_1 = M$ предикат p^n звуть *тотожньо істинним* (*хибним, невизначеним*) на множині M.

Визначення 1.11. Два предикати f^n і h^n визначені на сукупності M *рівносильні*, якщо їх множини істинності, хибності, невизначеності співпадають.

Наприклад, предикат $(x_1^2 + 2x_2^2 + 3x_3^2 + 4x_4^2 \geq 0)$ рівносильний на дійсній сукупності до предикату $(x_1^2 + x_2^2 + x_3^2 + x_4^2 \geq 0)$.

Розглянуті вище елементарні предикати визначені алфавіті T, тому до цих предикатів можливо застосувати звичайні операції логіки висловлювань $(\wedge^2, \vee^2, \neg^1, \Rightarrow^2, \Leftrightarrow^2)$.

Так, наприклад, якщо предикати $f(x)$ і $h(x)$ визначені на сукупності M, тоді і предикат $g(x) = f(x) \wedge h(x)$ визначений на $-$ M, причому множиною істинності $M_g \subseteq M$ предикату $g(x)$ буде спільна частина множини істинності предикатів f і h відповідно $M_f \subseteq M$ і $M_h \subseteq M$, тобто $M_g = M_f \bigcap M_h$. Аналогічно визначаються множини хибності та невизначеності предикату $g(x)$. Також можливо ввести предикат $q(x) = f(x) \vee h(x)$, для якого множина істинності $M_q = M_f \bigcup M_h \subseteq M$. В свою чергу до отриманих предикатів також можливо застосувати логічні операції. Так предикат $\neg g(x)$ є істинним для тих і тільки тих значень $x \in M$, для яких $-$ $g(x)$ є хибний.

Елементарні предикати і сконструйовані на їх основі більш складні предикати, без відносно множини визначеності (на вільних змінних), прийнято називати *формулами логіки предикатів*. Після підстановки у формулу логіки предикатів замість змінних конкретних предикатів, визначених на певній сукупності M, формула з вільними змінними стає конкретним предикатом на цій множині M.

Нехай F_1 і F_2 дві формули з множини формул F.

Визначення 1.12. Формули F_1 і F_2 логіки предикатів називаються *рівносильними на сукупності* M, якщо для будь якої підстановки конкретних предикатів, визначених на $-$ M, замість вільних змінних, ці формули утворюють рівносильність, за визначенням 1.11.

Не складно перевірити, що введене поняття рівносильності формул є відношенням еквівалентності, і тому множина формул F, визначених на сукупності M, може бути покрита класами еквівалентності. Окрім того для рівносильних формул логіки предикатів мають місце ті ж самі властивості рівносильних висловлювань. Наприклад, за законом де Моргана справедливо $\neg(F_1 \wedge F_2) \sim \neg F_1 \vee \neg F_2$.

Формальну систему на множині предикатів можливо побудувати за тією ж методикою, що і формальні системи логіки.

1.7.2. Завдання і вправи

1. Виписати декілька одно-місцині предикати, визначені на алфавіті $A = \{a, b, c\}$.
2. Визначити множину істинності предикатів, якщо x, y – дійсні числа:
   а) $x^2 - 4 = 0$,
   б) $x^2 > y$,
   в) $x^2 + 4 = 0$,
   г) $x^2 \geq y^2$.
3. Визначити множину хибності для предикатів вправи 2.
4. Визначити множину невизначеності для предикатів вправи 2.
5. Як зміниться множина істинності предикатів вправи 2, якщо їх помножити на m і а) $m > 0$; б) $m < 0$; в) $m = 0$.
6. Як зміниться множина хибності при умовах вправи 5?
7. Як зміниться множина невизначеності при умовах вправи 5?
8. Якій умові повинні задовольняти предикати f^n і h^n на сукупності M, щоб предикат $f^n \Leftrightarrow h^n$ був:
   а) тотожньо-істинним;
   б) тотожньо-хибним;
   в) тотожньо-невизначеним?
9. Якій умові повинні задовольняти предикати f^n і h^n на множині M, щоб предикат $f^n \Rightarrow h^n$ був:
   а) тотожньо-істинним;

б) тотожньо-хибним;
в) тотожньо-невизначеним?

10. Записати формулу: «через дві різні точки можливо провести тільки одну пряму».
11. Визначити одномісний предикат, як характеристику нечіткої величини.

1.8. Підсумковий коментар

Задачі викладання дисципліни «Програмна інженерія» поставлені у «Рекомендаціях» і частково вирішені стосовно проектування програм, їх тестування, аналізу і інше, наведені у навчальному посібнику [11] вимагають більш широкого підходу до змістовного питання викладання цієї дисципліни. Дисципліна за «Рекомендаціями» передбачає знання основ формальних систем, і вміння застосовувати їх при моделюванні алгоритмів та їх представлені у вигляді програм. Тому в наведеному розділі розглянуті базові питання формалізму, формальних систем, формальних структур, логічних числень, логічні перетворення та інших питань.

У розділі, як і в наступних розділах посібника розглянуто приклади і наведено багато завдань та вправ, які доповнюють викладені матеріали. Частина завдань може бути рекомендована викладачами для самостійного виконання студентами.

Зауважимо, що застосування формалізмів може бути виконане як при проектуванні, так і моделюванні алгоритмів та програм і їх логічних перетвореннях за наступною рекомендацією:

– задача,
– виділення її сутностей,
– формалізація,
– схема алгоритму,
– формальні перетворення,
– програма,
– формальні перетворення.

Фрагменти цієї схеми застосовуються при розв’язанні ряду завдань, розглянутих у додатках.

Література до розділу 1

1. *Глушков В. М., Цейтлин Г. Е., Ющенко Е. Л.* Алгебра. Языки. Программирование. – К.: Наукова думка, 1989. – 376 с.
2. История математики. Т1. М.: Наука, 1970. – 352 с.
3. *Клини С.* Математическая логика. – М.: Мир, 1973. – 480 с.
4. *Мальцев А.И.* Алгебраические системы. М.: Наука, 1970. – 391 с.
5. *Мандельсон Э.* Введение в математическую логику. М.: Наука, 1984 – 320 с.
6. *Новиков П.С.* Элементы математической логики. М.: Наука, 1973. – 400 с.
7. *Мартин – Лёф П.* Очерки по конструктивной математике. М.: Мир, 1975. – 136 с.
8. *Смальян Р.М.* Теория формальных систем. М.: Наука, 1981. – 208 с.
9. *Яблонский С. И., Гаврилов Г. П., Кудрявцев В. Б.* Функции алгебры логики и классы Поста. – М.: Наука, 1966. – 120 с.
10. *Post E.* Formal reductions of the general combinatorical decision problem. Am. J. Math. 65, – 1943. – P. 197-215.
11. *Бабенко Л.П., Лавріщева К.М.* Основи програмної інженерії: Навч. Посіб. – К.: Т-во “Знання”, КОО, 2001. – 269 с.

Розділ 2.
Елементи множинних структур та відношень при розробці моделей програм

У конструктивній математиці формальних систем принципово не використовується термін «множина» [12], але виходячи з природи програмування, введемо в обіг подальших викладень цей важливий термін. Предметна мова розділу складається з таких понять, як: *множина, підмножини, елементи множини, операції над множинами, нечіткі множини, конструктивні множини, кортежі, потужність множин, множинні об'єкти, відношення, відношення еквівалентності* та інше.

2.1. Мета розділу

За технологією програмування розробка програми починається із створення її моделі. Під моделлю програми розуміється спеціальний об'єкт в деякому відношенні її замінюючий. Об'єктом моделювання в програмній інженерії можуть бути окремі складові програми або програма в цілому. До окремих об'єктів моделювання в першу чергу слід віднести різні структури даних, наприклад, списки, масиви, черги, абстрактні типи даних тощо, котрі моделюються такими математичними об'єктами як множини, мультомножини та іншими. Моделювання програм може здійснюватися за допомогою елементів теорії систем, яка представляється множиною об'єктів програми (оператори, строки, процедури та інше) і відношеннями (зв'язками) на цій множині. Метою розділу є розширення понять різновидів множин і відношень для професійного моделювання окремих об'єктів і комплексів програм в програмній інженерії.

2.2. Поняття і способи визначення множини

В математиці як і в інших науках зустрічаються деякі первинні поняття які сприймаються на інтуїтивному рівні або визначені за допомогою аксіом. Ця невизначеність обумовлена методологічними причинами, узагальненістю поняття, суперечливістю і іншими причинами. До таких первинних понять має відношення поняття множини. Вийти з цього становища дозволяє описове визначення поняття множини.

2.2.1. Представлення множин

Наведемо описове визначення множини і її складових.

Визначення 2.1. Множиною зветься сукупність деяких однорідних вільних елементів.

Елементи множини є неподільними об'єктами (приймаються такими) і вільними по відношенню до порядку розташування у сукупності. Множини прийнято позначати великими літерами латинського або іншого алфавіту. Наприклад, позначимо через S – множину студентів університету, а через S_i – множину студентів i – групи університету. Для деяких множин визначеної природи прийняті позначення: $\mathbb{N}$ є множина натуральних чисел, $\mathbb{N}_0$ – множина натуральних чисел доповнена числом нуль, $\mathbb{R}$ – множина дійсних чисел і інші. Елементи множини будемо позначати малими літерами з індексами або без них. Щоб вказати склад множини, тобто з яких об`єктів складається множина, будемо використовувати фігурні дужки $\{,\}$. Якщо у вище наведеному прикладі позначити студентів i – групи університету як c_{ij}, $j = 1, 2, \ldots, 25$, тоді склад множини є $S_i = \{c_{i1}, c_{i2}, \ldots, c_{i25}\}$.

Зрозуміло, що кожен елемент множини може характеризуватися *ім`ям* або *ім`ям і значенням* (*властивістю*), які він має. Для того щоб позначати множину елементів, які мають властивість (значення) P – атрибут елемента, будемо використовувати такий запис $\{x; x \text{ має властивість } P\}$ або $\{{}_P x\}$ Наприклад, якщо студенти i – групи університету навчаються на відмінно, добре або задовільно, тоді можливо записати більш інформативну множину

$$\{c_{ij}; c_{ij} \text{ вчаться на } (\textit{відмінно} \mid \textit{добре} \mid \textit{задовільно}),\ j = 1, \ldots, 25\}$$

по відношенню до множини S_i; $\mathbb{N}=\{1,2,3,\ldots\}$ – означає, що множина $\mathbb{N}$ складається з елементів, які мають імена: 1,2,3 і так далі, і їх значення є числа 1,2,3,…, ця множина характеризується іменами елементів і їх значеннями в одному позначені; нехай A скінчений алфавіт з елементами a_i, тоді останній задається тільки множиною імен об`єктів (*символів*) $A=\{a_1,a_2,\ldots,a_n\}$. З наведених прикладів видно, що множини можуть складатися із скінченої кількості елементів і їх звуть *скінченими* в протилежному випадку – *нескінченними* множинами.

В класичній теорії множин [2] множина, яка не складається ні одного елемента і зветься *порожньою* множиною. Вона має позначення $\varnothing$, і конструктивно множна $\varnothing$ виникає, наприклад, при знаходженні перетину між двома множинами з різними елементами. Але у деяких випадках прикладних теорій таким чином введена порожня множина приводе до парадоксів. Наприклад, в теорії формальних граматик алфавіти можуть мати порожні символи, в схематології програм [8] множини програм можуть складатися з порожніх програм, конфігурації-множини алгоритмів Т`юрінга може складатися з порожніх символів, тому такі множини є порожніми за змістом , але не є $\varnothing$ – множинами у класичному розумінні. Щоб врахувати ці ситуації і уникнути подальших не порозумінь скористуємося результатами робіт [12, 13].

Визначення 2.2. Введемо в розгляд *порожній елемент,* який позначимо символом o і будемо вважати, що:

1) елемент o належить будь якій множині,

2) для порожньої множини $\varnothing$ має місце умова $\varnothing=\{o\}$.

В подальшому ми переконаємося, що задана таким чином множина $\varnothing$ не призводе до нових протиріч теорії множин.

Наведені деякі представлення множин не дають повної уяви про формалізм їх задавання. Введемо символи: $\in$ – *належить* і $\notin$ – *не належить*, тоді запис $x\in X$ означає: *x є елементом множини X* або *x належить до множини X*. Тепер множину S_i можливо записати таким виразом $\{c;c\in S_i\}$, в якому c – загальний елемент множини (загальний студент групи). Після точки з комою вказані властивості цього елементу $c\in S_i$, а запис $\{c;c\in S,c\notin S_i\}$ задає множину студентів університету за виключенням студентів i- групи, тобто $c\in S$ але $c\notin S_i$. Інколи множину можна задавати *діапазоном значень*, наприклад $\{x;x\in[-1:0.1:1]\}$, що відповідає одномірній послідовності значень від –1 до 1 з кроком 0.1. Множини можна задати і *функціональним* спосо-

бом, наприклад $\{y; y = arctg(x), x \in \mathbb{R}, -\pi/2 < x < \pi/2\}$ визначає множину елементів y, значення яких $V(y) = arctg(x)$. При умові, що дійсне значення x знаходиться між $-\pi/2$ і $\pi/2$. Інші способи визначення множин розглянуто у роботах [16, 17].

2.2.2. Підмножини

Множина може бути часткою іншої множини, якщо кожен об`єкт однієї множини є об`єктом другої множини. Щоб показати таку залежність використовують знаки *включення* ($\subset, \supset, \subseteq, \supseteq$). Тоді залежність по включенню відносно множин S і S_i можна записати як $S_i \subset S$ або $S \supset S_i$, це означає, що множина S_i є *підмножиною* множини S. Випадок включення $B \subseteq A$ означає, що множина B включається у множину A або ці множини співпадають (складаються з однакових об`єктів). Наведемо деякі властивості включень:

1) $A \subseteq A$ множина A є підмножиною самої себе;

2) $\varnothing \subseteq A$, тому що за визначенням 2.2 $o \in A$;

3) якщо виконуються включення $A \subseteq B$ і $B \subseteq C$, тоді $A \subseteq C$.

Визначення 2.3. Множини A і B звуть *рівними* (*однаковими*) $(A = B)$, якщо виконуються включення: $A \subseteq B$ і $B \subseteq A$.

Очевидні властивості рівності множин:

1) $A = A$;

2) якщо $A = B$, тоді $B = A$;

3) якщо $A = B$ і $B = C$, тоді $A = C$.

Якщо множина B є підмножиною множини A і A не є підмножиною множини B, тоді множину B звуть *власною підмножиною* множини A.

Твердження 2.1. Порожня множина $\varnothing$ є єдиною множиною.

▹ Дійсно, нехай $\varnothing_1$ і $\varnothing_2$ – дві порожні множини. Так як для будь якої множини A маємо, за другою властивістю включення, що $\varnothing \subseteq A$, тоді $\varnothing_2 \subseteq A$ і $\varnothing_1 \subseteq A$. Взявши в якості A множину $\varnothing_1$, отримаємо $\varnothing_2 \subseteq \varnothing_1$, а взявши в якості A множину $\varnothing_2$, отримаємо $\varnothing_1 \subseteq \varnothing_2$. Звідси маємо твердження $\varnothing_2 = \varnothing_1$. ◃

Об`єктами множини можуть бути множини. Множину усіх підмножин множини $A$ звуть *булеаном множини* $A$ і позначають $\mathrm{P}(A)$. Якщо об`єкти множини є однотипними множинами A_α тоді кажуть,

що задане *сімейство* множин $\{A_\alpha; \alpha \in I\}$, у якому I – множина індексів. Наприклад, якщо P_k – множина поліномів степені k з цілими коефіцієнтами, то сімейство поліномів є $\{P_k; k \in \mathbb{N}\}$.

На завершення цього параграфу розглянемо дві вправи.

Вправа 2.1. Необхідно задати множину інтервалів визначених на множині $\mathbb{R}$.

Спочатку визначимо об`єкти-інтервали [4, 5]. Інтервали на множині дійсних чисел можливо задавати декількома способами. Наприклад, як

$$A = [\underline{a}, \overline{a}] = \{x;\ x \in \mathbb{R},\ \underline{a} \le x \le \overline{a}\} \tag{2.1}$$

де числа $\underline{a}$ і $\overline{a}$ вказують на відстань від точки 0 на числовій вісі $\mathbb{R}$; або як

$$A = (m, \mp c) = \{x; x \in \mathbb{R}, m - c \le x \le m + c\}, \tag{2.2}$$

де число m задає відстань від точки 0 на числовій вісі $\mathbb{R}$, а число $c \in \mathbb{R}$ вказує значення відхилення від числа m Зрозуміло, що інтервал $A \subset \mathbb{R}$.

Якщо позначити через m середину інтервалу A, тобто $m = (\underline{a} + \overline{a})/2$, а через l – його довжину $l = \overline{a} - \underline{a} = 2c$, тоді очевидно, що вирази (2.1) і (2.2) визначають один і той же інтервал. Окрім того інтервал A можливо визначити і так $A = (m, l)$.

Між інтервалами A і B існує звичайне включення, при цьому інтервали співпадають $A = B$ тоді і тільки тоді, коли $\underline{a} = \underline{b}, \overline{a} = \overline{b}$. Інтервал A є виродженим, коли його кінці співпадають, тобто $\underline{a} = \overline{a} = a$ і інтервал A в цьому випадку ототожнюється з числом $a \in \mathbb{R}$. До речі, порожньому інтервалові відповідає $[o, o] = \varnothing$. Якщо тепер множину всіх інтервалів позначити через $\mathbb{I}(\mathbb{R})$, тоді можливо записати, що $\mathbb{R} \subset \mathbb{I}(\mathbb{R})$.

Вправа 2.2. На множині інтервалів визначити нечітку множину.

Розв'язок поставленої задачі суттєво залежить від *предметної області*, на якій визначена множина інтервалів. У якості предметної області візьмемо ринкову економічну систему, насичену множиною товарів $\{t_i; i \in M \subset \mathbb{N}\}$, ціни яких залежать від якості, місця продажу і іншого, тому товар t_i визначається інтервалом цін X_i. За допомогою характеристичної функції $\chi_i : X_i \to [0,1]$ [11] введемо показник «цінової ваги» товару, при цьому значення «цінової ваги» змінюється

від 0 до 1 і буде тим більшою, чим частіше, наприклад, ця ціна виставляється в торговельній мережі. Якщо характеристика χ_i опукла функція, така що

$$\chi_i(\lambda x_1 + (1-\lambda)x_2) > \min\{\chi_i(x_1), \chi_i(x_2)\}, \quad x_1, x_2 \in X_i \quad \text{і} \quad \lambda \in [0,1],$$

тоді пара $(x, \chi_i(x))$ зветься нечіткою ціновою величиною товару t_i, а набір таких *упорядкованих пар* (кожен елемент пари знаходиться на фіксованому місці) утворює *нечітку* цінову *множину* товарів

$$\tilde{X} = \{(x, \chi_i(x)); x \in X_i, i \in M\}.$$

У тому випадку, коли інтервал X_i вироджений, тобто $X_i = x$, характеристика чіткого елемента x має значення $\chi_i(x) = 1$. Тоді для *чіткої множини* цін x товарів характеристика $\chi(x)$ приймає значення $\chi(x) = \begin{cases} 1, & x \in X; \\ 0, & x \notin X \end{cases}$.

2.2.3. Завдання і вправи

1. Чи істинним є висловлювання:
   а) $\{a\} \subset \{\{a,b,c\},\{a,b\},a,c\}$;
   б) $\{a\} \subset \{b,\{a,c\},\{a\},\{c\}\}$;
   в) $a,b \in \{\{a,b\},a,c\}$;
   г) $a \in \{\{a\},\{a,b\},\{c\},b\}$;
   д) $\{a,b\} \subseteq \{\{a,b\},\{c\},a,b\}$?
2. Наведіть приклади таких множин A, B і C, для яких виконується третя умова включень і для яких ця умова не виконується, тобто при включеннях $A \subseteq B$, $B \subseteq C$ має місце $A \not\subseteq C$.
3. Довести істинність висловлювання $(A \subseteq \varnothing) \Leftrightarrow (A = \varnothing)$.
4. Для довільних множин A, B і C довести наступні висловлювання:
   а) $((A \subseteq B) \wedge (B \subseteq C)) \Rightarrow (A \subseteq C)$;
   б) $((A \subseteq B) \wedge (B \subset C)) \Rightarrow (A \subset C)$;

в) $((A \subset B) \wedge (B \subseteq C)) \Rightarrow (A \subset C)$;

г) $((A \subset B) \wedge (B \subset C)) \Rightarrow (A \subset C)$.

5. Навести всі елементи множини булеана $P(\varnothing)$.
6. Побудувати формальну систему над множиною інтервалів $\mathbb{I}(\mathbb{R})$.
7. Побудувати формальну систему над нечіткою множиною вправи 2.2.
8. Довести, що дві множини однакові тоді і тільки тоді, коли значення їх перетину і об'єднання співпадають.
9. Довести, що однаковість (=) множин – рефлексивна, симетрична і транзитивна.
10. Які множини утворюються наступними виразами? Які їх геометричні образи?

а) $\{(x, y);\ x = y^2,\ x, y \in \mathbb{R}\}$,

б) $\{(x, y);\ |x+5| = |3-y|,\ x, y \in \mathbb{R}\}$,

в) $\{(x, y);\ x = y^2 + x^2,\ x, y \in \mathbb{R}\}$,

г) $\{(x, y);\ 4 = y^2 + x^2, x = y,\ x, y \in \mathbb{R}\}$,

д) $\{(x, y);\ x \geq y^2,\ x, y \in \mathbb{R}\}$,

е) $\{(x, y);\ x + 4 \leq y^2,\ x, y \in \mathbb{R}\}$,

ж) $\{(x, y);\ x = y^2 - 2,\ x < 0,\ x, y \in \mathbb{R}\}$,

з) $\{(x, y);\ x < y^2, x > y,\ x, y \in \mathbb{R}\}$.

2.3. Множинні конструктивні об'єкти

Розглянемо *конструктивні* об'єкти, за допомогою яких можливо будувати інші об'єкти множин програмної інженерії.

2.3.1. Алфавіти

Найпростішим прикладом таких об'єктів є букви природної мови, з яких будуються слова, іншим прикладом конструктивних об'єктів є

вертикальна риска (|), за допомогою якої можливо побудувати множину натуральних чисел: {1 – |, 2 – ||, 3 – |||, ...}. Також набір із вертикальної (|), горизонтальної (−) та нахиленої (/) рисок, що дозволяє побудувати цілі та раціональні числа [12], або цифри поштових індексів. Природа конструктивного об'єкту різна (буква, знак, цифра, риска і інше), тому будемо використовувати уніфіковану назву імені об'єкту «*символ*». Таким чином

Визначення 2.4. Множину символів (елементів) будемо називати алфавітом.

Наприклад, для вище розглянутих прикладів $A=\{а,б,в,\ldots,я\}$ – алфавіт української мови, $B=\{|\}$ – алфавіт натурального ряду. В подальшому будуть розглядатися множини як алфавіти, котрі задовольняють наступним умовам:

- алфавіт складається з скінченої кількості символів, тобто він скінчений,
- порожній символ o завжди належить алфавітові, хоча не завжди виділяється в переліку його символів;
- символи алфавіту різні, тобто однакові символи в алфавіті не допускаються;
- елементами множини можуть бути різні комбінації символів цього ж алфавіту.

Згідно з умовами об'єкти алфавітів можуть бути *простими* і *сполученими*. Прості об'єкти неподільні, тобто не можуть бути сполучені з символів даного алфавіту. Сполучені ж об'єкти складаються з більше як одного символів алфавіту. Сполучення $oa=a$ будемо відносити до простих елементів. Наприклад, алфавіт $A=\{o,a,b,ab,aab\}$ містить в собі три простих елементи (символи) o,a,b і два сполучених елементи ab,aab. Наведемо декілька важливих визначень.

Визначення 2.5. Розміром алфавіту A зветься кількість його простих елементів (символів) і позначається $\dim A$.

Так для алфавіту $A=\{o,a,b,ab,aab\}$ його $\dim A=2$, $\dim\{o\}=\dim\varnothing=0$.

Визначення 2.6. Алфавіт зветься *базисним або стандартним*, якщо він складається тільки з символів, тобто усі його елементи прості.

Наприклад, двійковий алфавіт $B=\{a,b\}$ – стандартний.

Якщо $A\subseteq B$, тоді кажуть, що алфавіт A є підалфавітом алфавіту B і в разі $A\subset B$ кажуть, що A є власним під алфавітом алфавіту B.

Визначення 2.7. Алфавіти A і B звуть *рівними* $(A=B)$ $(A=B)$, якщо виконуються включення $A\subseteq B$ і $B\subseteq A$.

2.3.2. Конструктивні об'єкти

Конструктивні об'єкти є екземплярами структур даних в програмуванні. На заданому алфавіті можливо побудувати різні конструктивні об'єкти типи даних, строки програм тощо.

Нехай задано деякий алфавіт A.

Визначення 2.8. Словами (ланцюжками) над алфавітом A будемо називати конструктивні об'єкти, котрі побудовані за допомогою *індуктивного* процесу:

– слово $\varepsilon = o$ є порожнім словом;
– якщо конструктивний об'єкт l – слово над алфавітом A, тоді конструктивний об'єкт $l\alpha$, $\alpha\in A$ є також слово, побудоване над алфавітом A.

З визначення 2.8 маємо:

– що $l\varepsilon = \varepsilon l = l$, тобто порожнє слово виступає в ролі одиниці множини слів;
– якщо l слово над алфавітом A, тоді за індуктивним процесом $lll\ldots l = l^i$ також слово над алфавітом A;
– очевидно, $l^0 = \varepsilon$,
– над алфавітом $\{o,a,b,ab,aab\}$ можливо отримати нескінченну сукупність слів $\{\varepsilon,a,b,aa,bb,ab,\ldots\}$.

Множину слів побудованих над алфавітом A будемо позначати через $\mathbb{F}(A)$.

Нехай l слово над алфавітом A, тобто $l = a_1a_2\ldots a_k \in \mathbb{F}(A)$, тоді

Визначення 2.9. Довжиною слова l над алфавітом A є кількість символів, з яких складається це слово і позначають її так $|l| = k$.

Приймемо довжину порожнього слова ε за $|\varepsilon| = 0$. Зрозуміло, що будь яке не порожнє слово має довжину $k \geq 1$. Якщо l_1 і l_2 слова множини $\mathbb{F}(A)$, тоді для довжини слів мають виконуватися властивості

$$|l_1l_2| = |l_1| + |l_2| \quad \text{і} \quad |l_1^i| = i|l_1|. \tag{2.3}$$

Визначення 2.10. Два слова $l = b_1 b_2 \ldots b_m \in \mathbb{F}(B)$ і $q = a_1 a_2 \ldots a_n \in \mathbb{F}(A)$ *однакові* $l = q$, якщо $|l| = |q|$ або $n = m$ і $a_i = b_i$ для всіх значень індексів $1 \le i \le n$.

Таким чином ми визначили формальну систему $\mathbb{S}_A$ над алфавітом A, але побудовану за допомогою індуктивного (*рекурсивного*) правила $l = l\alpha, \quad \alpha \in A$, на основі якого побудована вільна мова $A^* = \mathbb{F}(A)$.

Якщо слова $l, q \in \mathbb{F}(B)$, то може статися, що слово l є частиною слова q, тобто $q = xly$, де $x, y \in \mathbb{F}(B)$, у цьому випадку слово l звуть *підсловом* (*підланцюжком*) слова q. Питання пов'язані з дослідженням повторних входжень підслів в слова множини $\mathbb{F}(B)$ розглядаються в проблемі Туе [13].

Розглянемо деяку конструктивну множину W це може бути алфавіт або будь яка множина слів.

Визначення 2.11. Упорядкована послідовність елементів (взагалі, різної природи) $w_i \in W, \quad i = 1, 2, \ldots, n$, тобто кожен елемент w_i займає $i-$ те місце у цій послідовності, зветься *упорядкованим кортежем* (кортежем) розміру n і позначається як $(w_1, w_2, \ldots, w_n)$.

Звертаємо увагу на те, що кортеж може бути неупорядкованим, але він відрізняється від множини неоднорідністю елементів. Якщо $n = 2$, тоді упорядкований кортеж (w_1, w_2) звуть *упорядкованою парою*. Вважається, що два кортежі $(w_1, w_2, \ldots, w_n)$, $(y_1, y_2, \ldots, y_m)$ співпадають, якщо вони мають однаковий розмір $n = m$ і $w_i = y_i, \quad i = 1, 2, \ldots n$. Очевидно, що кортежі $(w_1, w_2, \ldots, w_n)$ і $(w_2, w_1, \ldots, w_n)$ різні, а пара $(w, o) = (o, w) = w$ є *виродженою*. Кортеж порожній, якщо він утворений з порожніх елементів $(o, o, \ldots, o) = o$. Кортежі, як елементи можуть утворювати конструктивні множинні об'єкти [16, 17]. Частковим випадком упорядкованого кортежу є *список*, у якому об'єкти мають однакову природу.

Вправа 2.3. Конструктивно над двійковим алфавітом {0,1} побудувати вісімкову систему числення.

▹ У вісімковій системі числення кількість різних конструктивних об'єктів a_i дорівнює восьми, тобто $(a_1, a_2, \ldots, a_8)$.

Виходячи з того, що a_i повинні будуватися над двійковим алфавітом, тому довжина $|a_i| = 3$ $(2^3 = 8)$ для $i = \overline{1,8}$. Тепер процес побудови слів a_i відбувається за схемою алгоритму: спочатку набір $(a_1, a_2, \ldots, a_8)$ поділяється навпіл і першій групі набору приписується

символ 0, а другій – 1, як зображено на рис. 2.1, потім знову кожна з груп поділяється навпіл і першій приписується – 0, другій – 1 і так далі поки в кожній групі не зостанеться по одному об'єкту. Кодові слова вісімкової системи числення отримаємо, читаючи символи 0 або 1 спочатку процесу ділення (див. рис. 2.1). ◁

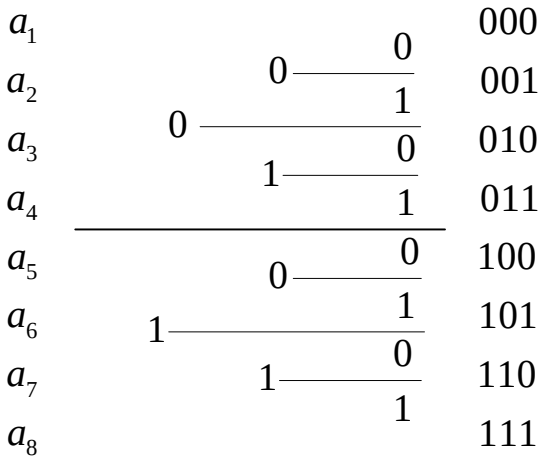


Рис. 2.1. Кодування вісімкової системи числення

Питання розглянуте у вправі 2.3 відноситься до проблем кодування інформації [3] при передачі даних в каналах зв'язку без завад.

Наведена на рис. 2.1 схема відтворює схему алгоритму ефективного кодування Шеннона, якщо a_i настають з однаковою ймовірністю.

Зрозуміло, що над множинами слів, кортежів можливо будувати більш складні формальні конструкції мультимножин, гібридних конструкцій-строк програм тощо.

2.3.3. Завдання і вправи

1. Створити конструктивний алфавіт для відтворення поштових індексів міст та населених пунктів України.
2. За допомогою яких конструктивних алфавітів можливо створити:
   а) множину інтервалів;
   б) множину кінцевих десятинних дробів;
   в) множину двійкових чисел?
3. Записати декілька раціональних чисел з застуванням алфавіту символів $\{-,|,/\}$.
4. Записати відомі символи функцій і вказати, які з них є простими, сполученими для заданого алфавіту.

5. Побудувати алфавіт на відомих автошляхових знаках, вказати прості символи.

6. Навести приклади базисних алфавітів розміру: 1; 2; 3; 4.

7. Для алфавіту $\{a, b, ab, c, d, cd\}$ вказати власні підалфавіти.

8. Для заданих алфавітів $A = \{o, a_1, a_2, \ldots, a_n\}$ і $B = \{o, b_1, b_2, \ldots, b_m\}$ знайти їх розміри.

9. Записати алфавіти мов програмування PASCAL і C.

10. Яка вільна мова може бути створена на алфавітові $B = \{0;1\}$?

11. Що собою уявляють кортежі у програмах записаних на мовах PASCAL і C?

12. Закодувати декілька слів у трійковому алфавітові $\{0,1,2\}$.

13. Закодувати декілька слів у двійковому алфавітові $\{-,|\}$.

2.4. Операції над множинами

Конструювання множинних об'єктів можна виконувати за допомогою операцій, деякі з яких розглядаються у цьому підрозділі.

2.4.1. Поняття операції і функціональної операції

Під *операцією* розуміють певну дію, яка виконується над об'єктами або над їх іменами (символами), або над їх значеннями – *операндами*. Кожна операція пов'язана з визначеною для неї кількістю *місць*. В залежності від природи операндів операцію називають *арифметичною*, *логічною*, *множинною* та ін. Операції прийнято позначати *знаками-символами*. Наприклад, відомі арифметичні операції: додавання, віднімання, множення і ділення позначаються символами $(+, -, \times, :)$. За кількістю місць операції визначається її *місцину (місність)*, наприклад, арифметичні операції двомісні і це позначається верхнім відповідним індексом, як $+^2, -^2, \times^2, (:)^2$.

Крім того можливо розглядати і «функціональні» операції визначені як функції (відображення, алгоритми). Наприклад, $\sin, \lg, \sqrt{\ }$ – одномісні функції; $\max, \min$ – k-місні функції. В загалі то, для позначення k-місних функцій використовують символьні позначення $f^k, h^k, \ldots$ або позначення з індексами $f_i^k, h_j^k, \ldots$. Слід відрізняти сим-

вольне позначення функції f^k від значення цієї функції так, якщо $t_1, t_2, \ldots, t_k$ – терми, тоді наступний вираз $f^k(t_1, t_2, \ldots, t_k)$ є також терм.

Визначення 2.12. Сигнатурою – Σ називають множину знаків-символів операцій (функцій, відображень) з врахуванням їх місцини

2.4.2. Правила виконання множинних операцій

Розглянемо множину операцій, яка задається сигнатурою $\Sigma = \left\{\cup^2, \cap^2, (\backslash)^2, \Delta^2, (')^1, \times^2\right\}$ і операції діють на підмножинах деякої множини D. Припустимо, що A, B і C – підмножини множини D. З'ясуємо за якими правилами діють ці операції.

Операція *об'єднання* $(\cup)$ визначається за формулою:

$$A \cup B = C, \quad C = \{x;\, x \in A \textit{ або } x \in B\},$$

тобто об'єднання двох множин A і B є множина C, елементи якої належать множині A або множині B. Якщо множини – алфавіти, тоді результуючий алфавіт звуть *розширеним алфавітом* $\overline{A}_B$ – читається, як «*алфавіт A розширений алфавітом B*». Наприклад, $A = \{a_1, a_2, a_3\}$, $B = \{b_1, b_2, a_1, a_2\}$, тоді розширення алфавіту A алфавітом B є $\overline{A}_B = \{a_1, a_2, a_3, b_1, b_2\}$.

Операція *перетину* $(\cap)$ двох множин визначається як

$$A \cap B = C, \quad C = \{x;\, x \in A \; i \; x \in B\},$$

тобто множина C складається із спільних елементів множин A і B. Для попереднього прикладу множин A і B їх перетин є $A \cap B = \{a_1, a_2\}$.

В тому випадку, коли A і B задовольняють умові $A \supset B$ розглядають операцію *різниці* $(\backslash)$, за якою

$$A \setminus B = C, \quad C = \left\{x;\, x \in A \; i \; x \notin B\right\} \cup \{o\}$$

множина C складається із елементів множини A за виключення елементів множини B і доповнюється порожнім елементом. Якщо множини A і B алфавіти, тоді різницю $A \setminus B$ називають *звуженням алфавіту A алфавітом B* і позначають $\underline{A}_B$. Наприклад, для введеного вище алфавіту A і $B = \{o, a_1, a_2\}$ маємо звуження $\underline{A}_B = \{o, a_3\}$.

Операція *симетричної різниці* (Δ) між множинами A і B визначається як комбінація операцій різниці та об'єднання за правилом

$$A\Delta B = (A \setminus B) \cup (B \setminus A).$$

Операція *доповнення* ($'$) підмножини A до підмножини D визначає її частку $D_1 \subset D$ так, що для кожного $x \in D_1$, $x \notin A$.

Декартовий добуток ($\times$) на множинах A і B визначає множину упорядкованих пар (a,b), тобто

$$A \times B = C, \quad C = \{(a,b); a \in A \, i \, b \in B\}.$$

На множині довільних кортежів розміру n визначається більш складний декартовий добуток множин $A_1, A_2, \ldots, A_n$

$$\begin{aligned}((\ldots(A_1 \times A_2) \times A_3) \times \ldots) \times A_n = A_1 \times A_2 \times \ldots A_n = \\ = \{(a_1, a_2, \ldots, a_n); a_1 \in A_1, a_2 \in A_2, \ldots, a_n \in A_n\}\end{aligned}.$$

У тому випадку коли всі множини $A_1, A_2, \ldots, A_n$ однакові, тобто $A_i = A$, їх декартовий добуток записується скорочено $A \times A \times \ldots \times A = A^n$.

Наведемо деякі властивості, яким задовольняють розглянуті операції.

1) $A \cup B = B \cup A$,
   $A \cap B = B \cap A$ – властивість комутативності.

2) $(A \cup B) \cup C = A \cup (B \cup C)$,
   $(A \cap B) \cap C = A \cap (B \cap C)$,
   $(A \times B) \times C = A \times (B \times C)$ – властивість асоціативності.

3) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$,
   $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$,
   $A \times (B \cup C) = (A \times B) \cup (A \times C)$,
   $A \times (B \cap C) = (A \times B) \cap (A \times C)$ – властивість дистрибутивності.

4) $(A \cup B)' = A' \cap B'$,
   $(A \cap B)' = A' \cup B'$ – властивість де Моргана.

5) $A \cup A = A$,
   $A \cap A = A$ – властивість ідемпотенції.

6) $A \cup A' = D$,
$A \cap A = A$ – властивість доповнення.

7) $A \cap D = A$.

8) $A \cup \varnothing = A$.

9) $A \times \varnothing = A$.

Класично вважається, що дві множини не перетинаються, якщо вони не мають спільних елементів крім порожнього елемента o, тобто $A \cap B = \varnothing$.

Як бачимо, введена конструктивним чином порожня множина, при уточнені класичної операції доповнення, не виводе за межі класичної теорії множин. Але, слід зауважити, що в класичній теорії множин має місце властивість $A \times \varnothing = \varnothing$, однак для пари $(a, o) = a$, яка відповідає цьому добуткові в нашому випадку не прийнятна, тобто $A \times \varnothing \neq \varnothing$, якщо $A \neq \varnothing$.

Вправа 2.4. На лінійній системі E над множиною дійсних чисел необхідно задати формальну систему інтервалів $\mathbb{S}_E$.

З'ясуємо спочатку, що розуміється під *лінійною системою*. Множина E зветься лінійною системою, якщо для кожних двох її елементів x і y визначена їх сума $x + y \in E$ і для будь якого елемента x та числа $a \in \mathbb{R}$ визначений добуток $ax \in E$, причому ці операції задовольняють наступним умовам:

1) $(x + y) + z = x + (y + z)$;
2) $x + y = y + x$;
3) у множині E існує такий елемент θ, що для будь якого $x \in E$ виконується $0x = \theta$, $0 \in \mathbb{R}$;
4) $(a + b)x = ax + bx$;
5) $a(x + y) = ax + ay$;
6) $(ab)x = a(bx)$;
7) $1 \cdot x = x$ $1 \in \mathbb{R}$.

▷ За умовою вправи елементи $x, y \in \mathbb{R}$, тому $\theta = 0$.

Розглянемо декартовий добуток E^2. Кожній упорядкованій парі $(x, y) \in E^2$ буде відповідати інтервал $[x, y]$, $x \leq y$.

До множини аксіом Λ віднесемо умови 1) – 7) та умову «для будь якого $x \in \mathbb{R}$».

Систему правил виводу інтервалів P задамо так:

$$\sigma = (\text{для будь якого } x \in \mathbb{R}),$$

$$\sigma_1 = ((x+y)+z = x+(y+z)),$$

...,

$$\sigma_7 = (1x = x),$$

$$\sigma \to x,$$

$$x \to [x-y, x+y],$$

$$x-y \to x+(-1)y,$$

$$y \to \sigma_i,$$

$$x \to \sigma_i,\ i = 1,2,...,7.\triangleleft$$

Визначена таким чином формальна система утворює симетричні інтервали і в свою чергу задає лінійну систему на множині інтервалів.

2.4.3. Завдання і вправи

1. Для алфавітів $A = \{o, a_1, a_2, \ldots, a_n\}$ і $B = \{o, b_1, b_2, \ldots, b_m\}$ знайти $\overline{A}_B$ і $\overline{B}_A$.
2. Для алфавітів $A = \{o\}$, $A_i = \{o, a_1, a_2, \ldots, a_i\}$, де $i = \overline{1,n}$. Знайти $\overline{A}_{A_i}, i = \overline{1,n}$; $\overline{A}_{i\,A_{i-1}}, i = \overline{2,n}$; $\overline{A}_{i-1\,A_i}, i = \overline{2,n}$.
3. Для алфавітів завдання 1, знайти $\underline{A}_B$ і $\underline{B}_A$.
4. Для алфавітів завдання 2, знайти $\underline{A}_{A_i}$, $\underline{A}_{i\,A}$, $i = \overline{1,n}$; $\underline{A}_{i\,A_{i-1}}$, $\underline{A}_{i-1\,A_i}$, $i = \overline{2,n}$.
5. Довести, що для будь яких множин A і B справедливі наступні співвідношення:
   а) $\varnothing \subseteq A \cap B \subseteq A \cup B$;
   б) $A \subseteq A \cup B$; $B \subseteq A \cup B$;
   в) $A \cap B \subseteq A$; $A \cap B \subseteq B$.
6. Для довільних множин A і B довести, що висловлювання $(A \cup B = \varnothing) \Leftrightarrow (A = B = \varnothing)$ – істинне.
7. Довести властивості операцій над множинами 1 – 9 (стор. 50-51).
8. Визначити операції $\cup, \cap, \setminus$ через операції:
   а) Δ і $\cup$;
   б) Δ і $\cap$.

9. Нехай задані множини A, B і C такі, що $B \subseteq A \subseteq C$. Розв'язати відносно множини X наступні системи рівнянь

$$\begin{cases} A \cap X = B, \\ A \cup X = C; \end{cases},$$

$$\begin{cases} A \setminus X = B, \\ A \cup X = C; \end{cases}.$$

10. Довести, що

$$\bigcup_{i \in I} \bigcup_{j \in J} A_{ij} = \bigcup_{j \in J} \bigcup_{i \in I} A_{ij},$$

$$\bigcap_{i \in I} \bigcap_{j \in J} A_{ij} = \bigcap_{j \in J} \bigcap_{i \in I} A_{ij}.$$

2.5. Множинна потужність

Важливою характеристикою множин, яка не залежить від природи її елементів, є *потужність*. Поняття потужності множин введено Г. Кантором у 1874 р.

2.5.1. Поняття і визначення потужності множин

Для скінченої множини A кількість її елементів приймається за потужність множини і позначається як $\#A$, в цьому випадку розмір і потужність множини співпадають але в загалі це не так.

Розглянемо дві множини A і B, якщо кожному елементу $a \in A$ за яким то правилом поставлений у відповідність об'єкт $f(a)$, тоді правило f називають функцією; якщо ж всі об'єкти $f(a) \in B$, тоді співставлення $f(a)$ називають також *відображенням з A в B*. В загалі, відображення f визначена на декартовому добутку, тобто $f \in A \times B$ і записується як $f : A \to B$. Відображення f множини A на множину B зветься *однозначним*, якщо будь яким двом елементам $a_1 \neq a_2 \in A$ відповідають різні об'єкти $f(a_1) \neq f(a_2)$ множини B. У тому випад-

ку, коли до відображення f існує *обернене відображення* $f^{-1}: B \to A$, яке також є однозначним, тоді кажуть, що відображення f є *взаємно однозначним* або *бієкцією*. Наприклад, відображення за правилом x^2, $x \in \mathbb{R}^+$ є взаємно однозначним в дійсній множині додатних чисел і не є таким на множині дійсних чисел $\mathbb{R}$.

Визначення 2.13. Множини A і B мають *однакову потужність*, якщо існує взаємно однозначне відображення A на B.

Розглянемо, наприклад, дві множини: $\{1,2,3,\ldots\}$ і $\{2,4,6,\ldots\}$, зрозуміло, що між їх елементами k і $2k$ існує взаємно однозначне відображення, якщо $k = 1,2,3,\ldots$. Таким чином, множина натуральних чисел $\mathbb{N}$ має однакову потужність з множиною усіх парних натуральних чисел.

Потужність множин в певному розумінні визначає їх «розмір». Так множина A зветься *скінченою*, якщо вона має потужність однакову з множиною $\{1,2,3,\ldots,n\}$, тобто $\#A = n$. Множина, яка не є скінченою, зветься *нескінченною*.

Визначення 2.14. Нескінченна множина, яка має однакову потужність з множиною натуральних чисел $\mathbb{N}$, зветься *переліченою множиною*.

Потужність множини натуральних чисел $\mathbb{N}$ прийнято позначати як $\aleph_0$.

Визначення 2.15. Множина, яка має потужність однакову з множиною дійсних чисел $\mathbb{R}$ зветься *континуальною*, або кажуть, що вона має потужність *континуума*. Множині $\mathbb{R}$ приписується потужність $\#\mathbb{R} = 2^{\aleph_0}$ або $\#\mathbb{R} = \aleph$.

У розглянутому вище прикладі множина парних чисел є переліченою, тобто її потужність є $\aleph_0$. А множина точок інтервалу $(-\pi/2, \pi/2)$ не є переліченою, виходячи з того, що відображення tgx є взаємно однозначним між множиною $(-\pi/2, \pi/2)$ і множиною $\mathbb{R}$, тобто заданий інтервал має потужність континуум – $\aleph$.

Питання відносно переліченості тієї чи іншої множини A, тобто побудови взаємно однозначного відображення множини A на множину $\mathbb{N}$ розв'язується [6]. Деякі питання переліченості множин винесені у вправи цього пункту.

Розглянемо декілька важливих для подальшого властивостей перелічених множин та встановимо одну особливість множини елементів вільної мови, побудованій на заданому алфавіті.

2.5.2. Властивості потужностей

Властивість 2.1. Будь яка підмножина переліченої множини скінчена або перелічена.

Властивість 2.2. Об'єднання переліченого сімейства перелічених множин – перелічене.

▹ Наведемо алгоритм доведення властивості 2.2. Для цього розглянемо сімейство перелічених множин $A_1, A_2, \ldots$ і нехай $a_{i1}, a_{i2}, \ldots$ – елементи множин $A_i \quad i = 1, 2, \ldots$ Тоді існує скінчена кількість елементів a_{ik}, для яких має місце $i + k = 2$; а також існує лише скінчена кількість елементів a_{ik}, за для яких $i + k = 3$ і так далі. Тепер перенумеруємо спочатку за зростанням індексу i всі елементи a_{ik}, для яких $i + k = 2$; потім за послідовністю наступних чисел пронумеруємо елементи, для котрих – $i + k = 3$, і так далі. Зрозуміло, що при такій нумерації кожен елемент a_{ik} має певний номер і різні елементи мають різні номери. Таким чином, об'єднання $\bigcup_{i \in \mathbb{N}} A_i$ – перелічене. ◃

Розглянемо деякий алфавіт $A = \{a_1, a_2, \ldots, a_n\}$ і вільну мову $\mathbb{F}(A)$ утворену на цьому алфавітові, тоді має місце

Твердження 2.2. Вільна мова $\mathbb{F}(A)$ – перелічена.

▹ Виберемо в мові $\mathbb{F}(A)$ спочатку всі ланцюжки довжини 1 і утворимо з них множину A_1, потім множину A_2 – з усіх ланцюжків мови довжини 2 і так далі. Таким чином побудована множина $\{A_1, A_2, \ldots\}$ є переліченою, окрім того ланцюжки кожної з множин A_k перелічені тому, що кількість її елементів дорівнює n^k. Зрозуміло, що вільна мова тепер може бути представлена як $\mathbb{F}(A) = \bigcup_{k \in \mathbb{N}} A_k$ і за властивістю 2.2 отримаємо результат твердження. ◃

2.5.3. Завдання і вправи

1. Для множин $A = \{a, b, c\}$ і $B = \{a, b, d, h, p\}$, знайти $C = A \bigcup B$ і $\#C$.

2. Показати, що для множини $\mathbb{N}_0 = \mathbb{N} \bigcup \{0\}$ її потужність $\#\mathbb{N}_0 = \aleph_0$. *Вказівка*: скористатися відображенням $\mathbb{N}$ у $\mathbb{N}_0$ за правилом $x \mapsto x - 1$ і довести бієкцію цього відображення.

3. Розробити алгоритм того, що множина $\mathbb{Z}$ цілих чисел перелічена.

Вказівка: скористатися відображенням $\mathbb{Z}$ у $\mathbb{N}_0$ за правилом $z \mapsto f(z)$, де $f(z) = \begin{cases} 2z+1, \text{ якщо } z \geq 0; \\ 2|z|, \text{ якщо } z < 0. \end{cases}$

4. Довести, що множина $\mathbb{Q}$ раціональних чисел має $\#\mathbb{Q} = \aleph_0$.

Вказівка: скористатися конструктивним алфавітом $\{-,|,\backslash\}$ для представлення $\mathbb{Q}$.

5. Довести, що множина $\mathbb{R}$ не перелічена.

6. Довести, що скінчена множина потужності k містить $\dfrac{k!}{(k-i)!\,i!}$ різних підмножин потужності $i \leq k$.

7. Нехай $[a,b]$ і $[c,d]$ інтервали числової вісі $\mathbb{R}$, знайти геометричні інтерпретацію множин:

а) $[c,d]\times[a,b]$;

б) $[a,b]^3$;

в) $[a,b]^4$.

8. Нехай множини A і B такі, що $\#A = k$ і $\#B = m$. Для яких значеннях k і m між множинами A і B існує взаємно однозначне відображення?

9. Встановити взаємно однозначне відображення між декартовими добутками A^k і A^r, якщо $r \in \{1,2,\ldots k\}$.

10. Довести, що можливо встановити взаємно однозначне відображення між множинами:

а) $A\times B$ і $B\times A$;

б) $A\times(B\times C)$ і $(A\times B)\times C$.

11. Нехай $f : A \to A$ взаємно однозначне відображення. Довести, що f^{-1} також взаємно однозначне відображення.

12. Довести, що, якщо з переліченої множини видалити скінчену кількість елементів, то отримаємо перелічену множину.

13. Довести, що, якщо всі множини сімейства A_i скінчені, попарно не перетинаються і не порожні, тоді множина $\bigcup_i A_i$ перелічена.

2.6. Відношення

Одним з основних понять математики у тому числі і конструктивної є відношення, яке застосовується для визначення зв'язків між деякими об'єктами. Теорія відношень є однією з гілок загальної алгебри [7,18] і роль відношень у сучасній математиці суттєво зросла після теоретико-множинної реконструкції проведеної у двадцятому столітті. З позицій відношень можливо трактувати звичайні алгебраїчні операції, функції та відображення, предикати математичної логіки і інші об'єкти математики та програмної інженерії. Предметна мова відношень охоплює такі поняття як: *об'єкти відношень*, *типи відношень*, *порядок відношень*, *місність відношень*, *операції над відношеннями*, *відношення еквівалентності*, *гомоморфізм відношень* та інші.

2.6.1. Поняття та визначення відношень

Розглядаючи ті чи інші об'єкти предметної області, виконуючи аналіз множин цих об'єктів тощо ми з'ясовуємо, що між ними існує деякий зв'язок, вони знаходитись в певному *відношенні*. Наприклад, букви: «а» і «б» українського алфавіту знаходяться у *відношенні порядку,* тобто буква «б» *слідує* за буквою «а»; значення чисел натуральної множини також знаходяться у відношенні порядку бо між будь якими числами $k_1, k_2 \in \mathbb{N}$ завжди можливо вказати, що k_1 «більше» за k_2, k_1 «менше» k_2 або k_1 «дорівнює» k_2. Для позначення відношень будемо використовувати грецькі символи з верхнім індексом, який вказує на *місність відношення*, або при потребі спеціальні знаки $(\rightarrow, >, <, =, \sim, \neq, \ldots)$, наприклад, двомісне відношення $\alpha^2 : y \rightarrow 2x;\ x = 1,2,3$ задане у вигляді формули, яка вказує на те, що цілим числам $y = 2,4,6$ ставиться у відповідність $(\rightarrow)$ числа $x = 1,2,3$. Для того, щоб вказати на яких конкретно множинах визначене відношення застосовують нижні індекси його позначки, так для попереднього випадку можливо записати $\alpha^2_{Y,X}$, де $Y = \{2,4,6\}$ і $X = \{1,2,3\}$. Причому звертаємо увагу на послідовність розташування індексів у відношенні, бо не завжди має місце $\alpha^2_{Y,X} = \alpha^2_{X,Y}$.

Нехай скінчена послідовність множин $A_1, A_2, \ldots, A_n$ така, що хоча б один її елемент не порожня множина і розглянемо декартовий добу-

ток на цих множинах $A_1 \times A_2 \times \ldots \times A_n$. Зрозуміло, що при зробленому припущенні цей добуток є не порожньою множиною.

Визначення 2.16. $n-$ місним відношенням, або коротше *n- відношенням ρ^n (відношенням порядку n)* на послідовності множин $A_1, A_2, \ldots, A_n$ зветься правило або закон за яким декартовому добутку $A_1 \times A_2 \times \ldots \times A_n$ ставиться у відповідність його частка. Отже відношенням $\rho^n_{A_1, A_2, \ldots, A_n}$ на послідовності $A_1, A_2, \ldots, A_n$ є підмножина декартового добутку $A_1 \times A_2 \times \ldots \times A_n$.

В подальшому, при потребі, замість виразу щодо відношення $\rho^n_{A_1, A_2, \ldots, A_n}$ будемо користуватися скороченим записом $\rho^n_{A_i}$. Включення $\rho^n_{A_i} \subseteq A_1 \times A_2 \times \ldots \times A_n$ означає, що елементи множин A_j, $j = 1, 2, \ldots, n$ знаходяться у відношенні $\rho^n_{A_i}$ і, якщо елементи $a_{ij} \in A_j$, $j = 1, \ldots n$, тоді послідовність $(a_{i1}, a_{i2}, \ldots, a_{in}) \in \rho^n_{A_i}$, де індекс $i = 1, 2, \ldots, n$.

2.6.2. Представлення і властивості відношень

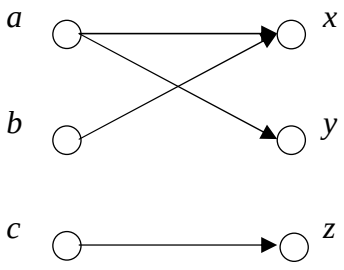
Відношення формальних систем можуть представлятися у вигляді таблиць, матриць, графіків, алгоритмів і інше. Наприклад, бінарне відношення $\rho^2_{A,B}$ на алфавітах $A = \{a, b, c\}$, $B = \{x, y, z\}$ можливо зобразити у вигляді однієї з форм, представлених на рис. 2.2.

$$\rho_{A,B} = \begin{pmatrix} a & x \\ a & y \\ b & x \\ c & z \end{pmatrix},$$

a

A/B	x	y	z
a	*	*	
b	*		
c			*

b



c

Рис. 2.2. Представлення відношень

Представлене на рис. 2.2, *a* відношення $\rho_{A,B}$ є матрицю суміжності, кожен стовпчик якої відповідає елементам множин, на яких визначене це відношення (перший відповідає множині A, другий – B). Табличне зображення відношення (див. рис. 2.2, *b*) вказує (зірочкою) які елементи множини A знаходиться у відношенні до елементами множини B.

Нескладно бачити, що площинне табличне зображення відношення зручно наводити для бінарних або тринарних відношень. Графічне зображення відношень (див. рис. 2.2, *c*) наглядне і не потребує пояснень, може бути застосоване для зображення відношень місцини $n \geq 2$. Доречи, бінарне відношення можливо записати у вигляді набору пар кортежів, наприклад, для відношень рис. 2.2 маємо:

$$\rho^2_{A,B} = \{(a,x),(a,y),(b,x),(c,z)\}.$$

В подальшому при зображені відношень $\rho^n_{A_j}$ ми будемо користуватися цими або іншими формами, причому інколи при матричній формі представлення будемо вказувати тільки $i-$ рядок цієї матриці за допомогою виразу $\rho^n(a_{i1}, a_{i2}, \ldots, a_{in})$.

Визначення 2.17. Перерізом відношення $\rho^n_{A_j}$ *по набору елементів множин* $A_1, A_2, \ldots, A_n$ і $A_{ij} = (a_{i1}, a_{i2}, \ldots, a_{ij-1}, a_{ij+1}, \ldots, a_{in})$ зветься множина $P^j_{A_{ij}}(\rho^n) = \{a_{ij};\ a_{ij} \in A_j\}$, а множина $A_j / \rho^n = \left\{P^j_{A_{ij}}(\rho^n)\right\}$ зветься *фактор-множиною* множини A_j по відношенню до $\rho^n_{A_i}$.

Для прикладу розглянемо відношення ρ^3 на алфавітах $A = \{a,b,c,d\}$ і $B = \{0,1\}$

$$\rho^3_{A,A,B} = \begin{array}{|c|c|c|c|c|} \hline & a & b & c & d \\ \hline a & & 1 & 0 & 1 \\ \hline b & 1 & & 1 & 1 \\ \hline c & 1 & 1 & & 0 \\ \hline d & 0 & 0 & 1 & \\ \hline \end{array} \qquad (2.4)$$

Переріз цього відношення по набору (a,b) є $P^3_{a,b}(\rho^3) = \{1\}$ і відповідно $P^2_{a,1}(\rho^3) = \{b,d\}$, а фактор-множина множини B по відношенню до $\rho^3_{A,A,B}$ задається виразом:

$$B/\rho^3 = \left\{P^3_{a,b}, P^3_{a,c}, P^3_{a,d}, P^3_{b,a}, P^3_{b,c}, P^3_{b,d}, P^3_{c,a}, P^3_{c,b}, P^3_{c,d}, P^3_{d,a}, P^3_{d,b}, P^3_{d,c}\right\} \qquad (2.5)$$

де $P^3_{a,b} = 1$, $P^3_{a,c} = 0$, $P^3_{a,d} = 1$, $P^3_{b,a} = 1$, $P^3_{b,c} = 1$, $P^3_{b,d} = 1$, $P^3_{c,a} = 1$, $P^3_{c,b} = 1$, $P^3_{c,d} = 0$, $P^3_{d,a} = 0$, $P^3_{d,b} = 0$ і $P^3_{d,c} = 1$.

Таким чином за допомогою фактор-множин маємо ще одну можливість визначати форму представлення відношень. Так фактор-множина B/ρ^3 однозначно задає відношення $\rho^3_{A,A,B}$ у формі (2.5).

Застосування відношень досить поширене в різних формальних теоріях. Наприклад, відношення встановлюють зв'язок: між операндами і операціями при побудові виразів, між виразами і операторами при записі програм і інше. На множині слів вільної мови $\mathbb{F}(A)$ над алфавітом A також можливо задати відношення, наприклад, за знаком $(<)$, таким чином, що для деяких слів $l_1, l_2 \in \mathbb{F}(A)$ це відношення має місце тоді і тільки тоді, коли $|l_1| < |l_2|$.

2.6.3. Операції над відношеннями і властивості операцій

Відношення є підмножинами множин, тому над ними виконуються звичайні теоретик o – множині операції. Ці операції також можливо трактувати як відношення певної місцини. Наприклад, операція декартового добутку як відношення відношень $\rho^k_{A_i}$ і $\lambda^m_{B_j}$ породжує відношення $\theta^{k+m}_{A_i,B_j} = \rho^k_{A_i} \times \lambda^m_{B_j}$.

Вправа 2.5. Розглянемо три алфавіти: $B = \{0,1\}$, $A = \{a_1, a_2\}$, $C = \{c_1, c_2, c_3\}$ і два відношення ρ^2, λ^3 на них:

$$\rho^2_{A,B} = \begin{pmatrix} a_1 & 0 \\ a_2 & 1 \end{pmatrix}, \quad \lambda^3_{A,B,C} = \begin{pmatrix} a_2 & 0 & c_1 \\ a_1 & 1 & c_2 \\ a_2 & 1 & c_3 \end{pmatrix}. \tag{2.6}$$

Необхідно знайти декартовий добуток відношень ρ і λ.

▹ Декартовий добуток цих відношень $\theta^5 = \rho^2 \times \lambda^3$ є

$$\theta^5_{A,B,A,B,C} = \begin{pmatrix} a_1 & 0 \\ a_2 & 1 \end{pmatrix} \times \begin{pmatrix} a_2 & 0 & c_1 \\ a_1 & 1 & c_2 \\ a_2 & 1 & c_3 \end{pmatrix} = \begin{pmatrix} a_1 & 0 & a_2 & 0 & c_1 \\ a_1 & 0 & a_1 & 1 & c_2 \\ a_1 & 0 & a_2 & 1 & c_3 \\ a_2 & 1 & a_2 & 0 & c_1 \\ a_2 & 1 & a_1 & 1 & c_2 \\ a_2 & 1 & a_2 & 1 & c_3 \end{pmatrix}. \text{ ◃}$$

Як видно з наведеної прикладу, відношення можуть мати досить складний вигляд, тому дослідження властивостей таких відношень може бути не простим. Для спрощення дослідження відношень застосується прийом «розщеплення» відношень на простіші за місциною.

Визначення 2.18. Прямою сумою двох відношень $\rho^k_{A_i}$ і $\gamma^{m-k}_{A_j}$ зі спільним останнім стовпцем матричного відношення ρ та першим стовпцем відношення – γ назвемо відношення $\lambda^m_{A_1,A_2,\ldots,A_m} = \rho^k_{A_1,A_2,\ldots,A_k} \boxplus \gamma^{m-k+1}_{A_k,A_{k+1},\ldots,Am}$, тобто відношення λ^m виникає тоді і тільки тоді, коли i рядки цих відношень мають вигляд $\rho^k(a_{i1},a_{i2},\ldots,a_{ik})$ і $\gamma^{m-k+1}(a_{ik},a_{ik+1},\ldots,a_{im})$. Таким чином для кожного рядка відношення λ^m має місце правило

$$\lambda^m(a_{i1},a_{i2},\ldots,a_{im}) = \rho^k(a_{i1},a_{i2},\ldots,a_{ik}) \boxplus \gamma^{m-k+1}(a_{ik},a_{ik+1},\ldots,a_{im}). \qquad (2.7)$$

Якщо A і B алфавіти вправи 2.5, а відношення $\rho^2_{A,B} = \begin{pmatrix} a_1 & 0 \\ a_2 & 1 \end{pmatrix}$ і $\gamma^3_{B,A,A} = \begin{pmatrix} 0 & a_2 & a_2 \\ 1 & a_2 & a_1 \end{pmatrix}$, тоді пряма сума відношень визначає відношення $\lambda^4_{A,B,A,A} = \begin{pmatrix} a_1 & 0 & a_2 & a_2 \\ a_2 & 1 & a_2 & a_1 \end{pmatrix}$.

Пряма сума як операція не комутативна, але асоціативна, тобто виконуються умови:

$$\rho^k \boxplus \gamma^m \neq \gamma^m \boxplus \rho^k, \qquad (2.8)$$

$$(\rho^k \boxplus \gamma^m) \boxplus \lambda^n = \rho^k \boxplus (\gamma^m \boxplus \lambda^n). \qquad (2.9)$$

Розглянемо важливий для подальшого результат.

Лема 2.1. Будь яке відношення місцини $n > 2$ може бути представлене у вигляді прямої суми бінарних відношень.

▹ Нехай задане відношення $\rho^n_{A_1,A_2,\ldots,A_n}$, тоді за індукцією правила (2.7) можливо записати для i - рядка відношення наступне

$$\rho^n(a_{i1},a_{i2},\ldots,a_{in}) = \rho^2(a_{i1},a_{i2}) \boxplus \rho^2(a_{i2},a_{i3}) \boxplus \ldots \boxplus \rho^2(a_{in-1},a_{in}),$$

що і доводить лему. ◃

Результат розглянутої леми вказує на те, що всяка множина може бути покрита її підмножинами за допомогою операції об'єднання, тобто можна припустити, що операції $(\cup)$ і $(\boxplus)$ і однакові. Але таке припущення хибне, виходячи з того, що операція об'єднання комутативна, а операція $(\boxplus)$ за умовою (2.8) не комутативна.

Наведемо деякі властивості бінарних відношень [7].

Якщо бінарне відношення ρ визначене на множинах A і B, і їх елементи $a \in A,\quad b \in B$; тоді замість запису $\rho(a,b)$ пишуть $a\rho b$.

Властивість 2.3. Бінарне відношення ρ *однозначнє* на множинах A і B, якщо для кожного елементу $a \in A$ існує один і тільки один елемент $b \in B$ який знаходиться у відношенні $a\rho b$.

Так бінарне відношення $\rho^2_{A,B}$ з вправи 2.5 – однозначне, а відношення $\gamma^2_{A,B} = \begin{pmatrix} a_1 & 0 \\ a_1 & 1 \\ a_2 & 0 \end{pmatrix}$ – неоднозначне: елемент a_1 знаходиться у відношенні з двома різними елементами.

Властивість 2.4. Бінарне відношення ρ *взаємно однозначним* на множинах A і B, якщо для кожного елементу $a \in A$ існує один і тільки один елемент $b \in B$; і навпаки для кожного елемента $b \in B$ існує один і тільки один елемент $a \in A$, які знаходиться у відношенні $a\rho b$.

Прикладом такого відношення є $\rho^2_{A,B}$ з вправи 2.5, відношення $\gamma^2_{A,B} = \begin{pmatrix} a_1 & 1 \\ a_2 & 1 \end{pmatrix}$ – взаємно неоднозначне, хоча є однозначним. Тривіальним випадком взаємно однозначного відношення є *тотожне відношення*, для якого довільний рядок має вигляд $\rho^2_{A,A}(a_i, a_i)$.

Властивість 2.5. Бінарне відношення ρ^{-1} є *оберненим відношенням* до відношення ρ на множинах A і B тоді і тільки тоді, коли $b\rho^{-1}a$.

Так для відношень *порядку* $(<,\leq)$ відповідають їм обернені відношення $(<)^{-1} = (>)$ і $(\leq)^{-1} = (\geq)$. Множину, елементи якої знаходяться у відношенні порядку $(<,\leq)$, або оберненого до нього за звичаєм називають *упорядкованою* [14].

Очевидно, бінарне відношення $\rho^2_{A,B}$ взаємно однозначне тоді і тільки тоді, коли відношення ρ і обернене до нього ρ^{-1} – однозначні, тобто для цього випадку вираз $\rho^2_{A,B} = \rho^2_{B,A}$ є істинним.

Тепер наведені властивості бінарних відношень можливо узагальнити на відношення більшої місцини.

Властивість 2.6. Відношення $\rho^n_{A_i}$ зветься однозначним, якщо його пряма сума складається з однозначних бінарних відношень.

Властивість 2.7. Відношення $\rho^n_{A_i}$ – взаємно однозначне, якщо його пряма сума складається з взаємно однозначних бінарних відношень.

Властивість 2.8. Множина $\rho^n_{A_i}$ – упорядкована, якщо її пряма сума складається з упорядкованих бінарних відношень.

Визначення 2.19. Відношення $(\rho^n_{A_i})^{-1}$ є оберненим до відношення $\rho^n_{A_i}$, якщо існує будь який рядок $(\rho^n)^{-1}(a_{in}, a_{in-1}, \ldots, a_{i1})$.

Теорема 2.1. Відношення $\rho^n_{A_i}$ має обернене відношення тоді і тільки тоді, коли для складових його прямої суми існують обернені відношення.

▹ Розглянемо відношення $\rho^n_{A_i}$, тоді за лемою 2.1 маємо

$$\rho^n(a_{i1}, a_{i2}, \ldots, a_{in}) = \rho^2(a_{i1}, a_{i2}) \boxplus \rho^2(a_{i2}, a_{i3}) \boxplus \ldots \boxplus \rho^2(a_{in-1}, a_{in}).$$

Нехай для кожного з бінарних відношень прямої суми існує обернене відношення, тоді побудуємо суму

$$\rho^{-1}(a_{in}, a_{in-1}) \boxplus \rho^{-1}(a_{in-1}, a_{in-2}) \boxplus \ldots \boxplus \rho^{-1}(a_{i2}, a_{i1}).$$

За визначеннями 2.18 і 2.19 отримаємо існування довільного рядка оберненого відношення $\rho^{-1}(a_{in}, a_{in-1}, \ldots, a_{i1})$.

Припустимо тепер, що існує обернене відношення $(\rho^n_{A_i})^{-1}$, тоді за лемою 2.1 для цього відношення запишемо

$$\rho^{-1}(a_{in}, a_{in-1}, \ldots, a_{i1}) = \rho^{-1}(a_{in}, a_{in-1}) \boxplus \rho^{-1}(a_{in-1}, a_{in-2}) \boxplus \ldots \boxplus \rho^{-1}(a_{i2}, a_{i1}).$$

Звідси маємо існування обернених бінарних відношень. ◃

Таким чином, для обернених відношень виконується властивість:

$$(\lambda^m_{A_1, A_2, \ldots, A_m} = \rho^k_{A_1, A_2, \ldots, A_k} \boxplus \gamma^{m-k+1}_{A_k, A_{k+1}, \ldots, Am}) \to ((\lambda^m_{A_m, A_{m-1}, \ldots, A_1})^{-1} =$$
$$= (\gamma^{m-k+1}_{A_m, A_{m-1}, \ldots, A_k})^{-1} \boxplus (\rho^k_{A_k, A_{k-1}, \ldots, A_1})^{-1}).$$

Для теоретико-множинних операцій також мають місце наступні властивості:

а) $(\gamma^k \subseteq \rho^k) \to ((\gamma^k)^{-1} \subseteq (\rho^k)^{-1})$,

б) $(\rho^k \times \gamma^m)^{-1} = (\gamma^m)^{-1} \times (\rho^k)^{-1}$,

в) $\left(\bigcup_{i=1}^{n} \rho_i^k\right)^{-1} = \bigcup_{i=1}^{n} (\rho_i^k)^{-1}$,

г) $\left(\bigcap_{i=1}^{n} \rho_i^k\right)^{-1} = \bigcap_{i=1}^{n} (\rho_i^k)^{-1}$,

д) $((\rho^k)^{-1})^{-1} = \rho^k$.

Окрім розглянутих вище дій над відношеннями існує велика кількість інших операцій, які можливо конструктивно ввести, так, наприклад, задати операції відображення і суперпозиції.

Якщо ρ^2 однозначне відношення визначене на алфавітах A і B, а відношення α^n визначене на декартовому добутку $A^k \times A_1^{k_1} \times A_2^{k_2} \times \times \cdots \times A_m^{k_m}$ такому, що $k + \sum_{i=1}^{m} k = n$. Тоді операція відображення $\rho^2 \alpha^n = \lambda^n$ визначає відношення на декартовому добутку $B^k \times A_1^{k_1} \times A_2^{k_2} \times \cdots \times A_m^{k_m}$ таке, що λ^n співпадає з відношенням α^n після заміни елементів алфавіту A на елементи алфавіту B за відношенням ρ^2. Розглянемо операцію відображення між відношеннями, які визначенні на алфавітах $A = (a,b,c,d,e)$, $B = (i,j,k,l)$, $C = (h,g)$ і $D = (t,s)$. Нехай:

$$\rho^2_{X,Y} = \begin{pmatrix} a & h \\ b & h \\ c & h \\ d & g \\ e & g \\ i & t \\ j & t \\ k & s \\ l & s \end{pmatrix}, \text{ де } X = \overline{A}_B,\ Y = \overline{C}_D; \qquad (2.10)$$

і $\alpha^3_{A,B,A}$, вигляд якого зручно представити таблицею

$$\alpha^3_{A,B,A} = \begin{array}{|c|c|c|c|c|c|} \hline B/A & a & b & c & d & e \\ \hline i & b & a & b & c & a \\ \hline j & a & b & c & b & c \\ \hline k & a & b & b & e & d \\ \hline l & b & c & a & e & e \\ \hline \end{array} \qquad (2.11)$$

Результатом операції відображення в цьому випадку є

$$\lambda^3_{D,C,C} = \rho^2_{X,Y}\alpha^3_{A,B,A} = \begin{array}{|c|c|c|c|c|c|} \hline D/C & h & h & h & g & g \\ \hline t & h & h & h & h & h \\ \hline t & h & h & h & h & h \\ \hline s & h & h & h & g & g \\ \hline s & h & h & h & g & g \\ \hline \end{array} .$$

Відображення λ можливо спростити видаливши в ньому однакові зв'язки і отримаємо:

$$\lambda^3_{D,C,C} = \begin{pmatrix} t & h & h \\ t & g & h \\ s & h & h \\ s & g & g \end{pmatrix}. \qquad (2.12)$$

Розглянемо операцію суперпозиції відображень. Операція суперпозиції поширена в математиці, так процес обчислення функції $\sin\sqrt{a^2+b^2}$ відбувається за послідовністю обчислень функцій: $(\cdot)^2$, +, $\sqrt{\ }$, sin; що записується суперпозиції обчислень $\sin(\sqrt{\ }(+((a)^2,(b)^2)))$.

Суперпозицію на відношеннях введемо за правилом [18].

Розглянемо відношення $\gamma^m_i, i=1,2,\ldots,n-1$ на множинах $A_1, A_2, \ldots, A_{m-1}, B_i$ і відношення ρ^n на множинах $B_1, B_2, \ldots, B_{n-1}, A_m$.

Визначення 2.20. Суперпозицією відношень $\rho^n, \gamma_1^m, \gamma_2^m, \ldots, \gamma_{n-1}^m$ зветься відношення місцини m $\lambda^m = \rho_{\gamma_i}^n$ на множинах $A_1, A_2, \ldots, A_m$ таке, що $i-$ рядок відношення λ^m має вигляд $\lambda^m(a_{i1}, a_{i2}, \ldots, a_{im})$ тоді і тільки тоді, коли знайдуться такі елементи $b_{ji} \in B_i$, $i = 1, 2, \ldots, n-1$, для яких має місце $\gamma_k^m(a_{i1}, a_{i2}, \ldots, a_{im-1}, b_{jk})$, $k = 1, 2, \ldots, n-1$; і $\rho^n(b_{j1}, b_{j2}, \ldots, b_{jn-1}, a_{im})$.

Пояснимо виконання операції суперпозиції на прикладі. Нехай на алфавітах: $A = \{a_1, a_2, a_3, a_4\}$, $B = \{b_1, b_2, b_3, b_4\}$ і $C = \{0, 1, 2, 3\}$ визначені відношення $\rho_{A,B}^2$, $\gamma_{B,C}^2$, $\lambda_{A,B,C}^3$ наступними виразами:

$$\rho_{A,B}^2 = \begin{pmatrix} a_2 & b_2 \\ a_3 & b_3 \\ a_4 & b_4 \end{pmatrix}, \quad \gamma_{B,C}^2 = \begin{pmatrix} b_1 & 0 \\ b_2 & 1 \\ b_3 & 2 \\ b_4 & 3 \end{pmatrix} \text{ і } \lambda_{A,B,C}^3 = \begin{pmatrix} a_1 & b_1 & 0 \\ a_2 & b_2 & 1 \\ a_3 & b_3 & 2 \\ a_4 & b_4 & 3 \end{pmatrix};$$

тоді суперпозиція $\lambda^3(\rho, \gamma)$ бінарних відношень ρ, γ є бінарне відношення $\beta_{A,C}^2 = \begin{pmatrix} a_2 & 1 \\ a_3 & 2 \\ a_4 & 3 \end{pmatrix}$ як результат «зчеплення» по алфавітові B у відношенні λ відношень ρ і γ.

2.6.4. Завдання і вправи

1. Знайти декартові добутки A^2, B^2, $A \times B$ і $B \times A$, якщо $A = \{1, 2\}$ і $B = \{3, 4, 5\}$

2. Довести, що існують такі множини A, B і C, для яких на декартовому добутку виконуються умови:

   а) $A \times B \neq B \times A$;

   б) $A \times (B \times C) \neq (A \times B) \times C$.

3. Вказати геометричну інтерпретацію множин:

   а) $A \times B$, де інтервали $A, B \in \mathbb{I}(\mathbb{R})$;

   б) A^2;

в) B^3;

г) $(A\times B)^2$.

4. На множині студентів, множині дисциплін, які вони вивчають і множині викладачів, які читають ці дисциплін визначити відношення. Якого порядку це відношення?

5. Для відношень ρ^2, знайти ρ^{-1}, область визначення відношення ρ $Dom\,\rho$ і область значень відношення ρ $Ran\,\rho$, де $Dom\,\rho=\{x;\ існує\ y\ таке, що\ (x,y)\in\rho\}$, $Ran\,\rho=\{x;\ існує\ y\ таке, що\ (y,x)\in\rho\}$:

а) $\rho=\{(a,b);\ a,b\in\mathbb{N}\ і\ a\ поділяє\ b\}$:

б) $\rho=\{(a,b);\ a,b\in\mathbb{N}\ і\ b\ поділяє\ a\}$;

в) $\rho=\{(a,b);\ a,b\in\mathbb{R}\ і\ a+b\geq 0\}$;

г) $\rho=\{(a,b);\ a,b\in R\ і\ a\leq 2b\}$

6. Для бінарних відношень $\alpha^2_{A,B}$ та $\beta^2_{A,B}$ і довільних елементів $a\in A$ та $b\in B$, довести:

а) $a(\alpha\cup\beta)b\Leftrightarrow a\alpha b\ або\ a\beta b$;

б) $a(\alpha\cap\beta)b\Leftrightarrow a\alpha b$ ł $a\beta b$;

в) $a\alpha' b\Leftrightarrow$ í ĺ $a\alpha b$.

7. Записати відношення у матричному вигляді, якщо вони задані наступними формульними виразами:

а) $y=3n;\ n=2,4,1$;

b) $y=2m^2;\ m=-2,-1,0,1,2$;

c) $y=n+2n;\ n=1,2,4$.

Які з них однозначні відношення?

8. Наступні відношення записати у формульному вигляді:

а) $\beta^2=\begin{pmatrix}1&1\\2&1\\3&1\end{pmatrix}$;

б) $\alpha^2 = \begin{pmatrix} 1 & 1 \\ 2 & 4 \\ 3 & 9 \end{pmatrix}$;

в) $\gamma^2 = \begin{pmatrix} 1 & 1 \\ 2 & 1/4 \\ 3 & 1/9 \end{pmatrix}$;

г) $\delta^2 = \begin{pmatrix} 1 & 10 \\ 2 & 9 \\ 3 & 8 \end{pmatrix}$;

д) $\varphi^2 = \begin{pmatrix} 1 & -2 \\ 2 & -1 \\ 3 & 0 \end{pmatrix}$;

е) $\phi^2 = \begin{pmatrix} 1 & -1 \\ 2 & -2 \\ 3 & -3 \end{pmatrix}$;

ж) $\varphi^2 = \begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{pmatrix}$.

Які з них однозначні і взаємно однозначні відношення?

9. Над відношеннями попереднього пункту виконати можливі операції відображення відношень та суперпозиції.

10. Нехай A і B базисні алфавіти такі, що $\#A = n$ і $\#B = m$:

а) скільки існує бінарних відношень між елементами цих алфавітів?

б) при яких значеннях n і m існують однозначні відношення між елементами алфавітів A і B?

в) при яких значеннях n і m існують взаємно однозначні відношення між елементами алфавітів A і B?

г) при яких значеннях n і m існують обернені відношення між елементами алфавітів A і B?

д) задати бінарні відношення на алфавітах A і B та знайти їх декартові добутки, перевірити комутативність добутків.

2.7. Відношення еквівалентності

У цьому параграфі розглянуто досить важливе відношення, за допомогою якого можливо розбити будь яку множину на класи і тим самим спростити дослідження цієї множини. Це може стосуватися множин програм, алгоритмів, формальних мов і інших формальних систем програмної інженерії.

2.7.1. Типи відношень еквівалентності

Відношення еквівалентності вводиться на бінарних відношеннях. Нехай бінарне відношення ρ визначене на деякій множині A.

Визначення 2.21. Відношення ρ на множині A зветься *рефлексивним*, якщо має місце $a\rho a$ для будь яких значень $a \in A$.

Наприклад, якщо на алфавіті $A = \{0,1,2\}$ визначені бінарні відношення α і β

$$\alpha_A = \begin{pmatrix} 0 & 1 \\ 0 & 2 \\ 1 & 2 \\ 2 & 0 \end{pmatrix} \text{ і } \beta_A = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{pmatrix}, \tag{2.13}$$

тоді відношення β – рефлексивне, а відношення α не є рефлексивним.

Визначення 2.22. Відношення ρ на множині A_A зветься *симетричним*, якщо виконується умова $a\rho b \to b\rho a$ для будь яких значень $a,b \in A$.

Зрозуміло, що для симетричного відношення завжди існує обернене відношення причому $\rho = \rho^{-1}$, тому відношення виразів (2.13): β – симетричне, а α – ні.

Визначення 2.23. Відношення ρ на множині A зветься *антисиметричним*, якщо виконується умова $(a\rho b \; i \; b\rho a) \to (a = b)$ для будь яких значень $a,b \in A$.

Очевидно, що відношення β виразу (2.13) є антисиметричним.

Визначення 2.24. Відношення ρ на множині A зветься *транзитивним*, якщо має місце $(a\rho b \; i \; b\rho c) \to a\rho c$ для будь яких значень $a,b,c \in A$.

Очевидно також, що відношення β виразу (2.13) є транзитивним.

Тепер в залежності від того, які властивості (за визначеннями 2.21 – 2.24) має відношення ρ можливо задати різні відношення еквівалентності. Введемо таке визначення еквівалентності.

Визначення 2.25. Відношення ρ на множині A зветься *відношенням еквівалентності*, якщо задовольняє визначенням 2.21, 2.22, 2.24, тобто, якщо воно є рефлексивним і симетричним, і транзитивним.

Посилаючись на визначення симетрії стверджуємо, якщо бінарне відношення ρ на множині A є відношенням еквівалентності, тоді і обернене відношення ρ^{-1} на множині A є також відношенням еквівалентності.

Визначення 2.26. Якщо відношення ρ на множині A є відношенням еквівалентності і деякий елемент $a \in A$, тоді *асоційована* з цим елементом підмножина $A_a = \{b \in A; a\rho b\}$ множини A зветься *класом еквівалентності.*

Елемент a асоційованої множини $A_a \subseteq A$ називають *представником класу еквівалентності.* Так для відношення еквівалентності β виразу (2.13) таких представників буде три елементи множини A, це 0, 1 і 2.

Тепер за визначеним відношенням еквівалентності можливо у множині A виділити класи еквівалентності A_a.

Теорема 2.2. Класи еквівалентності A_a за відношенні ρ розбивають множину A на множину множин тобто $A = \bigcup_a A_a$.

▷ За умовами теореми на множині A визначено відношення еквівалентності ρ_A^2, виділимо класи еквівалентності A_a для елементів $a \in A$. Доведемо спочатку, що серед множин A_a не порожніх класів. Дійсно, для $a \in A_a \subseteq A$ виконується умова $a\rho a$ і так як $a \neq o$ маємо клас $A_a \neq \varnothing$.

Покажемо що кожен клас є унікальним. Нехай A_a і A_b два класи еквівалентності, тоді або $A_a = A_b$, або $A_a \cap A_b = \varnothing$. Припустимо протилежне – існує елемент $c \in A$ такий, що $c \in A_a \cap A_b$ і $c \neq o$; тоді $c \in A_a$ і $c \in A_b$, тобто мають місце ствердження $a\rho c$ і $b\rho c$. Далі із властивості симетрії відношення еквівалентності ρ слідує, що $b\rho c \to c\rho b$, а за властивістю транзитивності отримаємо $(a\rho c \; i \; c\rho b) \to a\rho b$, тобто маємо наступне $b \in A_a$. Зрозуміло, що для

будь якого елемента $y \in A_b$ виконується умова $b \rho y$ і так як доведено, що умова $a \rho b$ має місце тоді за транзитивністю $(a \rho b \, i \, b \rho y) \to a \rho y$ маємо $y \in A_a$; звідки слідує наступне $A_b \subseteq A_a$. Аналогічно можливо довести, що має місце протилежне включення $A_a \subseteq A_b$, але такі включення можливі тільки, коли $A_a = A_b$. Таким чином множина A розбивається відношенням еквівалентності на різні класи. ◁

Вправа 2.6. Необхідно множину програм P, написаних на деякій мові розбити на класи еквівалентності.

▷ Кожна програма $p_i \in P$ або зациклюється, або зупиняється зі значенням $V(p_i)$. Для визначеності програми, які зациклюються будемо називати порожніми [14], тобто їх значення $V(p_j) = o$.

Введемо відношення еквівалентності програм. Дві програми $p_1, p_2 \in P$ – еквівалентні, якщо вони обидві або зациклюються ($V(p_1) = o$ і $V(p_2) = o$), або обидві зупиняються і повертають однакові значення $V(p_1) = V(p_2)$. Залишаючи читачеві перевірку того, що введене відношення є відношенням еквівалентності, з'ясуємо за теоремою 2.2, що кожен клас еквівалентних програм буде складатися з програм, які повертають одне і теж значення. Зауважимо, що буде тільки один клас у розбитті, в якому будуть знаходитися програми, які зациклюються (порожній клас). Тому в подальшому можливо розглядати множину програм яка утворена з представників класів еквівалентності, при цьому усі об'єкти множини будуть різні і ця множина буде мати один і тільки один порожній елемент. ◁

Визначення 2.27. Бінарне відношення ρ на множині A, яке задовольняє властивостям рефлективності і транзитивності і антисиметричності, зветься *відношенням часткового порядку*, а відповідно множина A – *частково упорядкованою*.

Розглянемо бінарне відношення включення $(\subset)$ на множині A і нехай A_i, A_j підмножини – A. Це відношення рефлексивне тому, що $A_i \subset A_i$. Якщо $A_i \subset A_j$ і $A_j \subset A_i$, то $A_i = A_j$, що визначає антисиметрію відношення; якщо ж $A_i \subset A_j$ і $A_j \subset A_k$, то $A_i \subset A_k$ і воно транзитивне. Таким чином відношення $(\subset)$ є відношенням часткового порядку.

Відношення $(\leq)$ і обернене до нього – $(\geq)$, як нескладно перевірити, є відношеннями порядку. Множина дійсних чисел $\mathbb{R}$, над яким діє це відношення є упорядкованою. На множині слів $\mathbb{F}(A)$ алфавіту A також

можливо задати частковий порядок, наприклад, наступним чином, нехай l_1 і l_2 довільні слова з множини $\mathbb{F}(A)$, тоді вони знаходяться у відношенні $\rho = (\leq)$, $l_1 \rho l_2$, тоді і тільки тоді, коли $|l_1| \leq |l_2|$.

Розглянемо поширення властивостей еквівалентності на відношення більшої місцини. Для цього, наприклад, можливо скористатися результатом леми 2.1 і введеними визначеннями властивостей бінарних відношень. Тоді для відношення місцини n введемо

Визначення 2.28. Відношення ρ^n є рефлексивним (симетричним, транзитивним), якщо відповідні бінарні відношення його прямої суми – рефлексивні (симетричні, транзитивні).

Наприклад, за введеним визначенням 2.28 наступні тринарні відношення $\alpha^3, \beta^3, \gamma^3$ визначені на трійковому алфавіті $A = \{a, b, c\}$:

$$\alpha^3 = \begin{pmatrix} a & a & a \\ b & b & b \\ c & c & c \\ a & b & c \end{pmatrix}, \quad \beta^3 = \begin{pmatrix} a & b & c \\ b & a & c \\ b & c & b \\ a & c & a \\ c & b & a \\ c & a & b \end{pmatrix}, \quad \gamma^3 = \begin{pmatrix} a & b & c \\ b & c & a \\ a & c & b \\ c & b & a \\ b & b & b \end{pmatrix};$$

мають властивості: α – рефлексивне, β – симетричне і β, γ – транзитивні.

Далі, аналогічно тому як це зроблено для бінарних відношень, вводиться відношення еквівалентності і класів еквівалентності для відношень місцини n.

2.7.2. Морфізми відношень

Як у природних і фізичних процесах є схожі явища, процеси тощо так і при розгляді різних відношень можливо з'ясувати їх схожість [8, 10, 13, 14]. Наприклад, два годинники один з яких відміряє секунди, другий хвилини схожі за виміром часу. В математиці це досягається за допомогою понять *гомоморфізму* (схожість за формою) і *ізоморфізму* (однаковість будови).

Визначення 2.29. Відношення α^n і β^n *гомоморфні*, якщо існує однозначне бінарне відношення ρ^2 таке, що $\rho^2 \alpha^n = \beta^n$; відношення ρ^2

у цьому випадку звуть гомоморфізмом. Якщо ж відношення ρ^2 взаємно однозначне, тоді відношення α^n і β^n звуть *ізоморфними*, а ρ^2 – *ізоморфізмом*, тобто для цього випадку має місце $\rho^2\alpha^n = \beta^n$ і $\alpha^n = (\rho^2)^{-1}\beta^n$. Нагадуємо, що під виразом $\rho^2\alpha^n$ розуміється операція відображення відношень, яка введена у попередньому пункті.

Наведені у попередньому пункті відношення (2.11) і (2.12) є гомоморфними, так як існує відношення (2.10), яке відношенню (2.11) ставить в однозначну відповідність відношення (2.12). А такі відношення α^3 і β^3 визначені на алфавітах $A=(a,b,c,d)$, $B=(g,h,i,j)$, $C=(e,f)$ і $D=(k,l)$, як

$$\alpha^3_{C,A,A} = \begin{array}{|c|c|c|c|c|}\hline C/A & a & b & c & d \\ \hline e & a & c & d & c \\ \hline f & b & a & d & c \\ \hline \end{array}, \quad \beta^3_{D,B,B} = \begin{array}{|c|c|c|c|c|}\hline D/B & g & h & i & j \\ \hline k & i & j & g & h \\ \hline l & i & h & g & g \\ \hline \end{array}$$

ізоморфні виходячи з того, що існує взаємно однозначне відношення

$$\rho^2_{X,Y} = \begin{pmatrix} f & k \\ e & l \\ c & g \\ a & h \\ d & i \\ b & j \end{pmatrix}, \text{ де } X = \overline{A}_C \text{ і } Y = \overline{B}_D.$$

За відношенням $\rho^2_{X,Y}$ згідно визначенню 2.29 відтворюється, як нескладно перевірити, відношення

$$\left(\rho^2_{X,Y}\right)^{-1}\beta^3_{D,B,B} = \begin{array}{|c|c|c|c|c|}\hline C/A & c & a & d & b \\ \hline f & d & b & c & a \\ \hline e & d & a & c & c \\ \hline \end{array},$$

котре з точністю до розташування рядків і стовпців співпадає з відношенням $\alpha^3_{C,A,A}$. Подальший аналіз відношень $\alpha^3_{C,A,A}$ і $\beta^3_{D,B,B}$ показує, що вони мають однакову будову, тобто їх не можна відрізнити. Дійс-

но, якщо скористатися позначками: $(a \to b)$ – односторонній зв'язок між елементами a і b, $(a \leftrightarrow b)$ – двосторонній зв'язок, $(\hookleftarrow a)$ – зв'язок елемента a на себе; їх схеми між елементних зв'язків можливо представити наступним чином.

Для відношення $\alpha^3_{C,A,A}$ –

e	f
$\hookleftarrow a \quad b$ $\downarrow$ $d \leftrightarrow c$	$a \leftrightarrow b$ $c \leftrightarrow d$

і для $\beta^3_{D,B,B}$ маємо

l	k
$\hookleftarrow h \quad j$ $\downarrow$ $i \leftrightarrow g$	$g \leftrightarrow i$ $h \leftrightarrow j$

.

Наведені схеми зв'язків однакові за формою будови цих відношень.

Твердження 2.3. Введене за визначенням 2.29 ізоморфне відношення (за операцією відображення відношень) задає відношення еквівалентності на множині відношень.

▷ Еквівалентне відношення є рефлексивним, симетричним і транзитивним. Доведемо, що ізоморфне відношення задовольняє цим трьом умовам. Нехай α^n і β^n ізоморфні відношення. Ізоморфне відношення: рефлексивне, тому що можливо вибрати у ролі ρ^2 тотожне відношення, за яким $\rho^2\alpha^n = \alpha^n$; симетричне через ізоморфність відношень α^n і β^n, тобто існує взаємно однозначне відношення ρ^2 таке, що $\rho^2\alpha^n = \beta^n$ і $(\rho^2)^{-1}\beta^n = \alpha^n$.

Нехай ρ_1^2 ізоморфізм ізоморфних відношень α^n і β^n, а ρ_2^2 ізоморфізм ізоморфних відношень β^n і γ^n. Для доведення властивості транзитивності ізоморфного відношення, необхідно встановити існування ізоморфізму ρ^2 між відношеннями α^n і γ^n. Припустимо, що відношення α^n, β^n та γ^n задані на відповідних множинах $A_i, B_i, C_i;\ i = \overline{1,n}$, а ізоморфізми $\rho_1^2(a_{ik}, b_{ij})$ – на множинах $A = \bigcup_{i=1}^{n} A_i$, $B = \bigcup_{i=1}^{n} B_i$ і $\rho_2^2(b_{ij}, c_{im})$ – на множинах $B, C = \bigcup_{i=1}^{n} C_i$, тоді відношення $\rho^2_{A,C}$ отримаємо як суперпозицію ізоморфізмів ρ_1^2 і ρ_2^2.

Очевидно, відношення ρ^2 є взаємно однозначним і є ізоморфізмом відношень α^n і γ^n. $\triangleleft$

Можливо виконати дослідження відношень іншим чином, наприклад, за допомогою упорядкованих кортежів або їх графічного представлення. Так рефлексивне відношення $a\rho a$ можливо зобразити у вигляді кортежу (a,a), взаємно однозначне відношення симетрії $a\rho b = b\rho a$ – у вигляді пари однакових кортежів $(a,b)=(b,a)$, відношення транзитивності: з $a\rho b$ і $b\rho c$ має місце $a\rho c$ можливо записати у вигляді формули на кортежах $(a,b)\wedge(b,c)\Rightarrow(a,c)$ та інше. Графічне представлення цих відношень зображено на рис. 2.3.

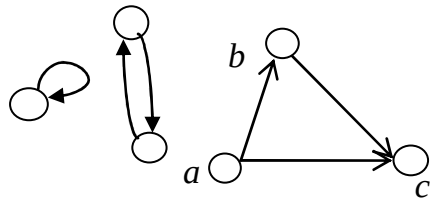


Рис. 2.3. Графічне представлення рефлексивного, симетричного і транзитивного відношень

Деякі інші приклади дослідження відношень на графах наведені у наступному розділі.

2.7.3. Завдання і вправи

1. Перевірити виконання властивостей для наступних бінарних відношень:
   - «a кратне b» у множині цілих чисел;
   - «пряма a має спільні точки з прямою b» у множині прямих на площині;
   - «коло a дотикається до b» у множині кіл на площині;
   - «a початковий символ слова b» у вільній мові $\mathbb{F}(A)$.
2. Довести, що для рефлексивних відношень ρ_1 і ρ_2 рефлексивні і відношення: $\rho_1 \cup \rho_2$, $\rho_1 \cap \rho_2$ і $\rho_{1,2}^{-1}$.
3. Нехай відношення $\rho = (\cup)$ визначене на множинах A і B. Чи завжди таке відношення є відношенням еквівалентності? Наведіть контр приклад.
4. Побудувати бінарне відношення:
   - рефлексивне і симетричне, і не транзитивне;
   - рефлексивне і антисиметричне, і не транзитивне;

- рефлексивне і не симетричне, і транзитивне;
- не рефлексивне і не симетричне, і транзитивне;
- не рефлексивне і симетричне, і транзитивне;

5. Показати, що наведені відношення є відношеннями порядку:
   - «*a* важче *b*» у множині деталей;
   - «*a* підлеглий *b*» у множині посад;
   - «*a* старший за віком *b*» у множині людей;
   - «*a* не перевищує *b*» у множині номерів будинків вулиці;
   - «*a* настає з *b*» у множині висловлювань.

6. Нехай ρ бінарне відношення між ланцюжками l_1 і l_2 деякої вільної мови завдається висловлюванням «l_2 частина l_1». З'ясувати чи відношення ρ є упорядкованим, чи ні?

7. Нехай множина M складається з чотирьох символьних змістовних слів української мови таких, що кожне з них загальний іменник у називному відмінку. Відношення ρ толерантне, якщо у відношенні $l_1 \rho l_2$, слова $l_1, l_2 \in M$ і відрізняються одне від одного не більше як на одну букву; тоді відповідні слова l_1, l_2 звуть толерантними. Створити з «мухи – слона», тобто побудувати на множині M послідовність толерантних слів, початком якої є слово «муха», а кінцевим словом «слон».

8. Довести ізоморфність відношень:

$$\alpha^2 = \begin{pmatrix} a & c \\ b & c \\ c & d \\ d & d \\ e & b \end{pmatrix} \text{ і } \beta^2 = \begin{pmatrix} p & r \\ q & q \\ r & q \\ s & p \\ t & r \end{pmatrix}$$

9. Побудувати схеми зв'язків цих відношень.

10. Знайти всі ізоморфізми для наведених відношень:

$$\alpha^2 = \begin{pmatrix} a & b \\ b & c \\ c & a \end{pmatrix} \text{ і } \beta^2 = \begin{pmatrix} p & r \\ q & p \\ r & q \end{pmatrix}.$$

11. Створити схеми будови наведених у попередньому завдані відношень. Представити схеми зв'язків цих відношень у вигляді кортежів і зобразити їх графічно.

2.8. Підсумковий коментар

У цьому розділі розглянуто фундаментальні поняття множин та відношень, як складових інструментарію моделювання і розробки програм. Зокрема, розглянуті класичні множини, їх способи завдань, операції над підмножинах, властивості операцій. Розробка структур даних програм і деяких інших конструкцій штучних систем призвели до окремого опису елементів конструктивних множин таких, як алфавіти, вільні мови та інше.

Моделювання штучних систем таких, наприклад, як бази даних, бази знань не обходяться без елементів відношень, яким у розділі приділена увага. Наведені відомості про способи задавання відношень їх властивостям та розглянуто спектр операцій над відношеннями. Введене поняття еквівалентності і морфізмів над відношеннями надає досить зручний інструмент перетворень в системах штучного інтелекту.

Програмування також не може обійтися без поняття відношення. Донедавна, з формальної точки зору, програма – множина зв'язаних операторів, тобто операторів, які знаходяться у відповідному відношенні. Але сучасне представлення програми за Кнутом [15] це мультимножина зв'язаних строк, так як строки в програмі можуть повторюватися. Це ж стосується і строк, в яких елементи можуть повторюватися. Наше [16, 17] представлення моделі сучасної програми і її строк, як зв'язана гібридна мультимножина, котра складається із мультимножин, мультимножиних списків у різних комбінаціях та іншого.

Поглиблене знайомство з розглянутих питань можна отримати в роботах [1, 2, 8, 9, 10, 13, 18].

Література до розділу 2

1. *Босов А. А.* Функции множества и их применение. – Дніпродзержинськ: Видавничий дім «Андрій», – 2007. – 186 с.
2. *Бурбаки Н.* Теория множеств. – М.: Физматгиз, 1965. – 455 с.
3. *Дмитриев В.И.* Прикладная теория информации. – М.: Высш. шк., 1989. – 320 с.

4. *Жолен Л., Кифер М., Дидри О., Вальтер Э.* Прикладной интервальный анализ. – М.:Ижевск: Институт компьютерных исследований, 2007. – 468 с.
5. *Калмыков С. А., Шокин Ю. И., Юлдашев З. Х.* Методы интервального анализа. Новосибирск: Наука, 1986. – 221 с.
6. *Клини С.* Математическая логика. – М.: Мир, 1973. – 480 с.
7. *Кострикин А. И.* Введение в алгебру. – М.: Наука, 1977. – 496 с.
8. *Котов В. Е., Сабельфельд В. К.* Теория схем программ. М.: Наука, 1991. – 248 с.
9. *Куратовский К., Мостовский А.* Теория множеств. М.: Мир, 1970. – 446 с.
10. *Марков А. А., Нагорный Н. М.* Теория алгорифмов. – М.: Наука, 1984 – 432 с.
11. Нечеткие множества и теория возможностей. Последние достижения. / Под ред. Р.Р. Ягера. М.: Радио и связь, 1986. – 408 с.
12. *Мартин – Лёф П.* Очерки по конструктивной математике. М.: Мир, 1975. – 136 с.
13. *Саломаа А.* Жемчужины теории формальных языков. М.: Мир, 1986. – 159 с.
14. *Скорняков Л. А.* Элементы теории структур. 2-е изд., доп. М.: Наука, 1982 – 160 с.
15. *Knuth D.* Context-Free Multilanguages // Theoretical Studies in Computer Science. – Academic Press, 1992. – P. 1 – 13.
16. *Ильман В. М., Шинкаренко В. И.* Конструктивное представление множественных объектов и их свойства. // НАНУ, ІПС. Проблемы программирования. – 2014. – № 1. – С. 3-17.
17. *Босов А. А., Ильман В. В., Халипова Н. В.* Множественные объекты. / Наука та прогрес транспорту. Вісник Дніпропетр. нац. ун-ту заліз. трансп, 2015, № 3. – С. 145 – 161.
18. *Глушков В. М., Цейтлин Г. Е., Ющенко Е. Л.* Алгебра. Языки. Программирование. – К.: Наукова думка, 1989. – 376 с.

Розділ 3.
Методи аналізу програм на основі графових моделей

3.1. Мета розділу

У другому розділі посібника було звернуто увагу на можливість системного моделювання програм за допомогою графів. Виходячи з цього розглянуто елементи теорії графів, їх форми представлення, властивості, перетворення тощо і запропоновані деякі методи аналізу програм та схем програм. Предметна мова розділу використовує поняття: *вершина графу*, *площинний граф*, *орієнтований та неорієнтований графи*, *схеми програм*, *виміри графу* та інші.

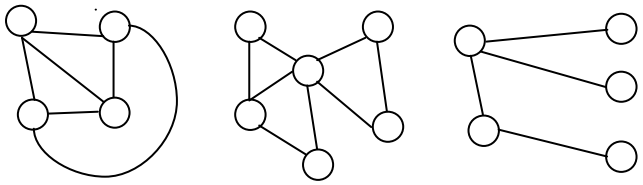
3.2. Загальні поняття графів

У попередньому розділі було показано, що відношення можуть задаватися графічно. Розглянемо спосіб формального визначення графа.

Визначення 3.1. Скінченим графом G зветься упорядкована пара множин C і D, $G = \langle C, D \rangle$, де $C = \{c_i; i \in I\}$ деякий скінчений алфавіт, який назвемо множиною *вершин* і – D множина відношень на цій множині вершин [12]. Множину C також звуть носієм графу.

Найбільшого застосування набули *площинні* графи визначенні на бінарних відношеннях. Характерним для таких графів є те, що множина відношень D залежить від типів елементів. Якщо множина D формується на списках (кортежах) декартового добутку C^2, тоді граф називають *орієнтованим* (*орграфом*), якщо D утворено на неорієнтованих парах елементів множини C^2, тоді граф звуть *неорієнтова-*

ним. Кортежі $(c_i, c_j) \in C^2$ графу задають відношення – *дуги* (вершина c_i є *початком* дуги, а вершина c_j – *кінцем* дуги) і – утворюють орграф, а неорієнтовані пари (c_i, c_j) формують *ребра*. Очевидно, множина D визначається множиною пар (c_i, c_j). Ребро або дуга може починатися і закінчуватися в одній і тій же вершині такі ребра або дуги звуть *петлями*. Площинні графи зручно зображати графічно. Наприклад, неорієнтовані графи зображено на рис. 3.1.



a b c

Рис. 3.1. Приклади неорієнтованих графів

Геометричні образи блок-схем алгоритмів відтворюють орграфи, вершинами яких є умовні, безумовні та інші типи блоків, зв'язані дугами, які вказують на послідовність виконання дій за цими блоками. Так орграф на рис. 3.2 зображає блок-схему алгоритму знаходження $\max\{a, b, c;\ a, b, c \in \mathbb{R},\ a \neq b \neq c\}$.

Вершини графу (c_i, c_j) може бути з'єднана декількома ребрами або одно направленими дугами. Такі ребра (дуги) називають *кратними*. Граф, в якому допускаються кратні ребра зветься *мультографом*, а якщо ж допускаються і петлі, то – *псевдографом*.

Вважається, що дві різні вершини $c_i, c_j \in C$ є *суміжними*, якщо між ними існує хоча б одне відношення, тобто вони повинні бути з'єднані хоча б одним ребром або хоча б однією дугою. Відповідно два ребра, які мають спільну вершину звуть також *суміжними*. Зокрема, для графів за суміжними вершинами можна утворити *ланцюжки*.

Визначення 3.2. Степеню $st(c)$ вершини $c \in C$ зветься кількість ребер усіх вершин суміжних до c.

Так степені вершин графу представленого на рис. 3.1, *а* однакові і дорівнюють трьом.

Степені вершин графів задовольняють наступним властивостям [10, 11, 12]:

1) кожна петля вершин збільшує її степінь на два,

2) сума степенів усіх вершин графу завжди парна,

3) кількість вершин непарних степенів графу парна.

и послідовності *степенів вершин* та інше.

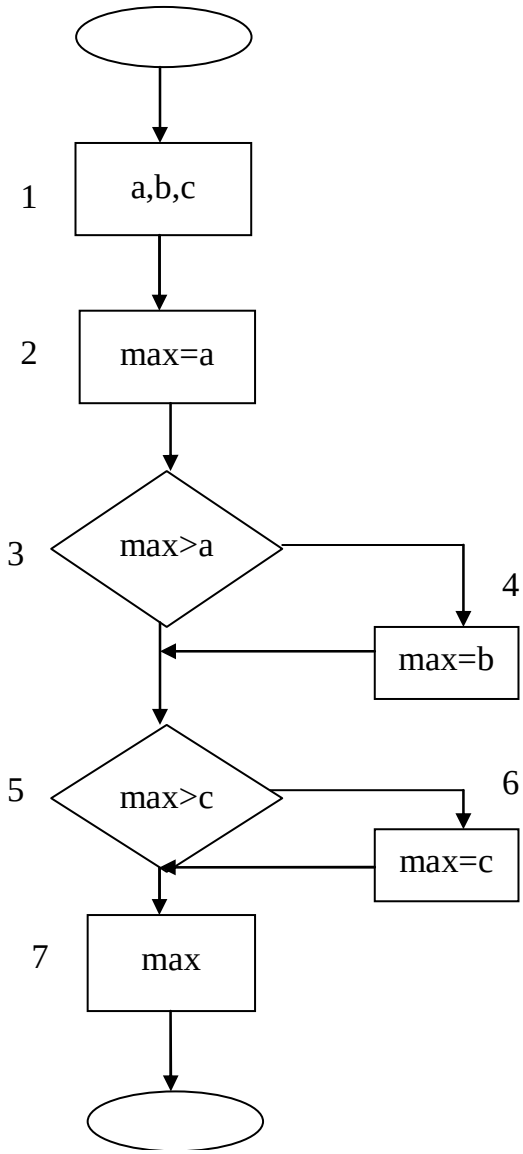
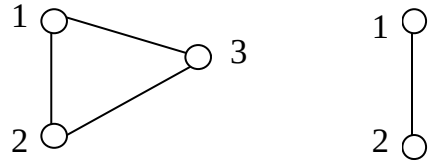


Рис. 3.2. Блок-схема алгоритму знаходження максимального значення трьох чисел

Послідовність суміжних пар вершин $\{c_i, c_j\}, \{c_j, c_k\}, \ldots, \{c_n, c_m\}$ графу G яку позначимо $l = (c_i, c_j, c_k, \ldots, c_m)$ утворює його *ланцюжок* з *початком* у вершині c_i і *кінцем* у вершині c_m. Ланцюжок, в якому немає повторень вершин звуть *простим ланцюжком* якщо у такому ланцюжкові вершини початку і кінця співпадають тоді ланцюжок l називають *циклом*. Граф, який складається тільки з простих ланцюжків зветься *неорієнтованим деревом.* Спільна вершина декількох ланцюжків дерева утворює його *корінь*. Дерева можуть бути одно або багато кореневі. Для дерев характерно, що степінь їх внутрішніх вершин c_k є $st(c_k) \geq 2$, для кінцевих вершин c_m – $st(c_k) = 1$.

Визначення 3.3. Граф $G_1 = \langle C_1, D_1 \rangle$ є *підграфом* графу $G = \langle C, D \rangle$, якщо $C_1 \subseteq C$ та $D_1 \subseteq D$ і це позначається $G_1 \subseteq G$.

Наприклад, якщо в графові G на рис. 3.3, *a* видалити вершину 3 і суміжні з нею ребра, то отримаємо підграф G_1, зображений на рис. 3.3, *b*.



a) b)

Рис. 3.3. Граф і його підграф

Для представлення графів у вигляді комбінацій їх підграфів застосовують операцію *об'єднання* графів $(\cup^2)$. Нехай задані графи $G_1 = \langle C_1, D_1 \rangle$ і $G_2 = \langle C_2, D_2 \rangle$, тоді їх об'єднанням є граф $G = G_1 \cup G_2$ такий, що $C = C_1 \bigcup C_2$ і $D = D_1 \bigcup D_2$, де $(\bigcup^2)$ теоретико-множинна операція об'єднання. Отже об'єднання двох графів рис. 3.3 дає граф зображений на рис. 3.3, *a*.

Визначення 3.4.. Множина підграфів $\{G_i\}$ (за виключенням власного підграфу $G_j = G$) графу G зветься *утворюючою* [6] для цього графу, якщо $G = \bigcup_i G_i$.

Наприклад, для графу зображеного на рис. 3.3, *a* утворюючою множиною може бути сукупність графів утворених парами $(1,2)$, $(2,3)$ і $(1,3)$. Зрозуміло, що утворююча множина підграфів неоднозначна, так для розглянутого прикладу утворююча множина може складатися з двох підграфів побудованих на множинах пар $\{(1,2),(1,3)\}$ і $\{(2,3)\}$.

В разі потреби дуги або ребра графів помічають символами якогось скінченого алфавіту $E = \{e_k; k \in K\}$. У такому випадку графи називають *навантаженими* (дивися позначки «+» та «–» над дугами на рис. 3.2) і позначаються такі графи G як $\langle C, D, E \rangle$. Для навантажених графів характерним є те, що будь яка дуга (ребро) (c_i, c_j), навантажена символом $e_k \in E$, може бути представлене у формі $e_{k,i,j}$ [7]. Тобто запис $e_{k,i,j}$ означає, що дуга (ребро) з позначкою ваги e_k з'єднує вер-

шини c_i і c_j деякого графу G. Тому інколи замість висловлювання «дуга графу (c_i, c_j)» стверджують «дуга графу $e_{k,i,j}$», або скорочено – «дуга графу e_k», якщо вершини графу не суттєві.

Розглянемо деякі алгебраїчні поняття та властивості площинного орграфу $G = \langle C, D, E \rangle$.

Прийнято вважати дуги з позначками $e_1, e_2 \in E$ *суміжними*, якщо вони мають спільну вершину $c \in C$. Зрозуміло, що для суміжних дуг e_1 і e_2 вершина c може бути початком будь якої дуги або дуг, а також кінцем дуги або цих дуг. Кажуть, що дуга з позначкою e *виходить* з вершини c, якщо ця вершина є початком дуги e і – *входить* у вершину c, коли ця вершина є кінцем дуги e.

Визначення 3.5. Степені вершини переноситься і на орграфи, так кількість суміжних дуг (вхідних і вихідних) вершини c називається степінню вершини c $(st\, c)$. Вершину c, для якої $st(c) = 0$ звуть *ізольованою вершиною.*

Очевидно, ізольованій вершині c графу G відповідає неупорядкована пара $(c, o) = (o, c)$. Нагадуємо, що o – порожній елемент алфавіту C і у алфавітові E також є порожній елемент – *порожня дуга*, тому парі (c, o) відповідає порожня дуга алфавіту E.

3.3. Формульне представлення графу

Представлення графу у графічній формі не зручно при аналітичних і машинних перетвореннях, тому пропонується аналітичний спосіб його запису.

Визначення 3.6. Послідовність дуг e_{s_k, i_k, j_k} графу G

$$P = \{e_{k_1, i_1, j_1}, e_{k_2, i_2, j_2}, \ldots, e_{k_m, i_m, j_m}\} \tag{3.1}$$

назвемо *шляхом довжини* m, тоді і тільки тоді коли $c_{j_r} = c_{i_{r+1}} \neq o$ для всіх r, $1 \le r \le m-1$. При цьому вершини c_{i_1} і c_{j_m} є *початком* і *кінцем* шляху, а проміжні вершини назвемо *внутрішніми* вершинами шляху P.

У випадку коли початок шляху (3.1) співпадає з його кінцем, тоді такий шлях звуть *циклом* або *контуром*. Частковим випадком циклу при його довжині $r = 1$ є *петля*, тобто петлі відповідає дуга (c, c).

Очевидно, послідовність позначок дуг шляху P_k утворюють *ланцюжки* $l_k = e_{k_1} e_{k_2} \dots e_{k_m}$, побудовані над алфавітом E, тобто $l_k \in \mathbb{F}(E)$. Нехай ланцюжки l_1 і l_2 утворені шляхами P_1 і P_2. Шлях P_1 є підшляхом шляху P_2 $(P_1 \subseteq P_2)$, якщо ланцюжок l_1 є підланцюжком ланцюжка l_2, тобто $l_1 \subseteq l_2$.

Нехай $\{P_i;\ i \in I\}$ – множина всіх різних шляхів графу G. Якщо ввести позначку «або» $(\vee)$, за допомогою якої записувати розгалуження шляхів графу, тоді граф G можливо представити у формі

$$G = \underset{i \in I}{\vee} P_i . \tag{3.2}$$

Так, наприклад, графи з розгалуженнями зображені на рис. 3.4 за представленням (3.2) можна записати: $(e_{1,1,2}) \vee (e_{2,1,2})$ див. рис. 3.4, *a*, $(e_{1,1,2}, e_{2,2,3}) \vee (e_{1,1,2}, e_{3,2,4})$ – рис. 3.4, *b* і $(e_{1,1,2}, e_{2,2,3}) \vee (e_{3,4,2}, e_{2,2,3})$ – рис. 3.4, *c*.

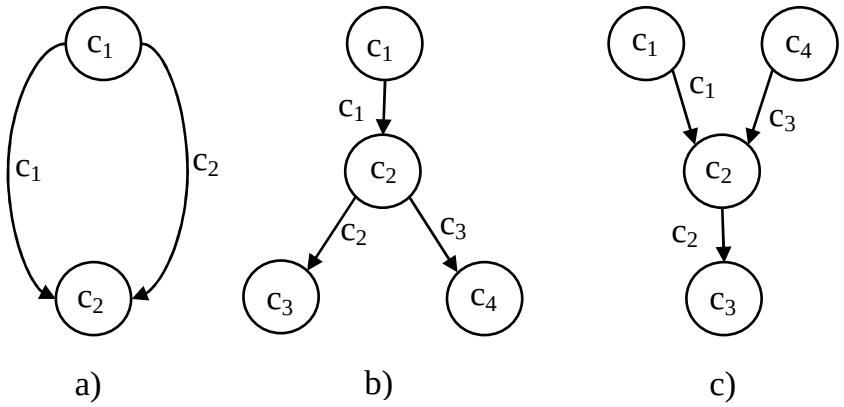


Рис. 3.4. Навантажені графи

Припустимо, що вершини c_i і c_j зв'язані декількома шляхами P_k, котрі утворюють відповідні ланцюжки l_k. Для графу G можливо визначити *відстань* $\rho(c_i, c_j)$ між вершинами c_i і c_j за формулою:

$$\rho(c_i, c_j) = \min_k \left| l_k \right| \tag{3.3}$$

та *діаметр* графу

$$d(G) = \max_{i,j} \rho(c_i, c_j) . \tag{3.4}$$

Функція відстані ρ на парах (c_i, c_j) графу G визначає метрику графу і для неї виконуються метричні властивості:

1) $\rho(c_i, c_j) = 0$ тоді і тільки тоді коли $c_i = c_j$;

2) $\rho(c_i, c_j) = \rho(c_j, c_i)$;

3) $\rho(c_i, c_j) + \rho(c_j, c_k) \geq \rho(c_i, c_k)$.

Визначення 3.7. Шлях P графу G називається *маршрутом* якщо не один з його підшляхів не утворює циклів або петель.

Для маршрутів характерним є те, що кожна його внутрішня вершина c_j на маршруті має $st(c_j) = 2$.

За допомогою маршрутів також можливо визначити дерево.

Визначення 3.8. Граф, який складається тільки з маршрутів і має для них одну спільну вершину утворює дерево.

Є багато різновидів дерев, так дерева кожна внутрішня вершина c, яких має $st(c) = 3$ звуть *бінарними деревами*, при $st(c) = 4$ – *тринарними деревами*.

Визначення 3.9. Вершини c_1 і c_2 графу G є *зв'язаними,* якщо вони утворюють петлю $(c_1 = c_2)$ або існує маршрут, для якого одна з них є початком, а друга кінцем.

Зв'язаність вершин графу визначає відношення, котре має властивості: рефлективності, симетричності та транзитивності. Відношення визначає порядок на вершинах графу і, окрім того, за цим відношенням множина вершин C розбивається на класи еквівалентності C_{c_i} попарно зв'язаних вершин. Граф, усі вершини якого попарно зв'язані, зветься *сильно зв'язковим.* Зрозуміло, що граф з ізольованими вершинами не є зв'язковим. Зазначимо, що зв'язність графу дозволяє скорегувати поняття ізольованої вершини. До ізольованих вершин відносяться такі вершини c_i, які не зв'язані з іншими вершинами c_j, $j \neq i$ графу. Разом з тим, наприклад, ізольована вершина c_i з петлею (c_i, c_i) є зв'язаною і її $st(c_i) = 2$.

Зв'язність графу досить важлива характеристика графів через яку визначаються структури графів, їх кількісні показники та інші алгебраїчні параметри графів. Дослідити зв'язність графу можливо через застосування матриць суміжностей $\{m_{ij}\}$, де $m_{ij} = \begin{cases} 1, & (c_i, c_j) \in D, \\ 0, & (c_i, c_j) \notin D, \end{cases}$ якщо

граф не навантажений і для навантаженого графу

$$m_{ij} = \begin{cases} e_k,\ e_{k,i,j} \in G, \\ 0,\ (c_i, c_j) \notin D. \end{cases}$$

Наприклад, для графу зображеного на рис. 3.2 матриця суміжностей відтворює зв'язки його вершин (1, 2, 3, 4, 5, 6, 7):

$$M = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & - & + & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & - & + \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}.$$

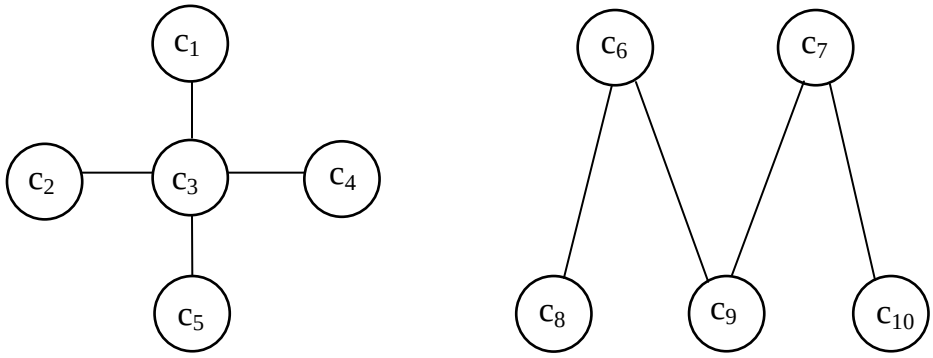
Елементи цієї матриці у першому рядку вказують на те, що петля на вершині 1 відсутня і вона зв'язана дугою з вершиною 2 в напрямку від вершини 1 до вершини 2, і інших вершин суміжних з нею немає. Аналогічно можна дослідити зв'язаність графу за іншими рядками матриці M.

3.4. Морфізми і перетворення графів

Вершини графу розташовуються при зображені довільним чином. Через це при дослідженні графів виникає необхідність з'ясувати питання однакової будову (*структури*) графів. Раніше вказувалося, що властивість однакової будови декількох множин можливо визначити за допомогою понять *гомоморфізму* і *ізоморфізму* – однакова форма, однаковий вид. Застосуємо поняття морфізму до графів.

Визначення 3.10. Графи $G_1 = \langle C_1, D_1, E_1 \rangle$ і $G_2 = \langle C_2, D_2, E_2 \rangle$ називають *гомоморфними* (*ізоморфними*), якщо між їх вершинами та їх дугами можливо встановити (взаємно) однозначне бінарне відношення, при якому зберігається орієнтація дуг.

Графи у ізоморфному класі неможливо відрізнити за структурою і тому можливо виділити одного представника класу для подальших досліджень. Наприклад, зображені на рис. 3.5 неорієнтовані графи деякого класу – ізоморфні тому, що між парами вершин: (c_1, c_6), (c_5, c_7), (c_2, c_8), (c_3, c_9) і (c_4, c_{10}) існує взаємно однозначне відношення. Отже представником цього класу може бути граф рис. 3.5, *a* або рис. 3.5, *b*.



a) b)

Рис. 3.5. Ізоморфні графи

Зауважимо, що з'ясування ізоморфізму графів можна виконати за формалізованими схемами суміжностей графів [16, 17].

Однаковість графів за структурою можна застосувати для з'ясування структурної еквівалентності програм, що важливо для аналізу програм і спрощення програмних комплексів.

Вважається, що дві програми $\Re_1$ і $\Re_2$ структурно еквівалентні, якщо їх структурні графи G_1 і G_2 ізоморфні, як не орієнтовані і направлення дуг у відповідних вершин цих графів співпадає.

Наведемо процедуру алгоритму для встановлення ізоморфізму структурних графів G_1 і G_2 [4] програм $\Re_1$ і $\Re_2$,

1) для програм $\Re_1$ і $\Re_2$ будуються структурні графи, за вершини яких приймаються відповідні оператори програм (вершини графів мають різні унікальні позначки, наприклад, $\acute{n}_i$ для першого графу і u_j для – другого), котрі з'єднуються дугами за семантикою програм;
2) визначаються степені вершин графів $st(c_i)$ і $st(u_j)$, на яких утворюються класи вершин з однаковими степенями $\acute{N}_k^s$ і U_r^g відповідно для графів G_1 і G_2, де s і g – степені вершин класів;

3) якщо кількості класів графів не однакова або кількості вершин у класах $\acute{N}_k^s$ і U_r^g з однаковими степенями різна, тоді робиться висновок, що графи не ізоморфні – у іншому випадку процес дослідження продовжується;
4) вважається, що вершина $\acute{n}_i$ графу G_1 взаємно однозначна до u_j графу G_2, якщо серед класів $\acute{N}_k^s$ і U_r^g знайдуться такі, для яких $s = g$ і вони містять по одній вершині, тому виконується перевірка на існування таких класів і відповідних вершин; для класів з більшою кількістю вершин необхідно перейти до наступного пункту;
5) виконується процес розбиття класів з більшою кількістю вершин ніж одна на підкласи доти поки у підкласах не зостанеться по одній вершині, якщо це можливо;
6) якщо розбиття не можливе, тоді при наявності у відповідних класах альтернативних вершин на взаємно однозначну відповідність, приймається рішення про ізоморфність графів, якщо альтернативності вершин відсутня – графи не ізоморфні.

Детально процес розбиття на підкласи наведено в додатках при розгляді задачі еквівалентності програм.

Поняття ізоморфізму можливо поширити на дуги одного і того ж графу. Дуги (c_i, c_j) і (c_k, c_s) графу G ізоморфні, якщо між ними можливо встановити взаємно однозначне бінарне відношення зі збереженням орієнтації дуг. Така залежність між дугами графу G зветься *автоморфізмом*. Очевидно, петлі графу G автоморфні, а маршрути графу G складаються з автоморфних пар (класів) вершин. Виходячи з автоморфізму дуг маршруту довжини n, довжину маршруту можливо скоротити до одиниці при цьому позначивши дугу словом $(e_j e_k \ldots e_n)$. Зрозуміло, що такі скорочення маршрутів дають можливість спрощувати форму графа і відповідно спрощувати подальше їх дослідження.

Наведемо деякі правила перетворення графів, за якими можна спростити навантажений граф. Для спрощення запису перетворень, правила перетворень демонструються на фрагментах шляхів [6 – 8].

1. Правило *скорочення шляху*:

$$(e_{s,k_1,k_2}, e_{j,k_2,k_3}, \ldots, e_{r,k_{m-1},k_m}) = ((e_s e_j \cdots e_r)_{k_1,k_m}),$$

якому після скорочення відповідає ланцюжок $l = (e_s e_j \cdots e_r)$.

2. Правило *виключення паралельної дуги*:

$$e_{1,i,j} \vee e_{2,i,j} = (e_1 \vee e_2)_{i,j},$$

при якому вершини зберігаються. Якщо $i = j$, то маємо частковий випадок правила – виключення паралельних петель.

3. Правило *виключення альтернативної дуги*:

$$e_{1,i,j} \vee e_{2,i,k} = \begin{cases} (e_1 \vee e_2)_{i,j}, \textit{ вершина } k - \textit{виключена}; \\ (e_1 \vee e_2)_{i,k}, \textit{ вершина } j - \textit{виключена}; \end{cases}$$

за яким дві суміжні дуги замінюються однією – (c_i, c_j) або (c_i, c_k).

4. Правило *виключення петлі*:

$$(e_{1,i,i}, (e_{2,i,j} \vee e_{3,i,k})) = (e_1^n e_2)_{i,j} \vee (e_1^n e_3)_{i,k},$$

де e_1^n – ланцюжок довільної довжини, утворений петлею $e_{1,i,i}$.

Правило поширюється і на заключну петлю альтернативних шляхів, тобто

$$\left(\left(e_{1,i,k} \vee e_{2,j,k} \right), e_{3,k,k} \right) = \left(e_1 e_3^n \right)_{i,k} \vee \left(e_2 e_3^n \right)_{j,k}.$$

5. Правило *виключення контурів*:

$$(e_{1,i,j}, e_{3,j,i}) \vee e_{2,i,k} = ((e_1 e_3)^n e_1)_{i,j} \vee ((e_1 e_3)^n e_2)_{i,k},$$

яке узагальнює правило 4.

6. Правило *виключення вершини*:

$$\begin{aligned} &(e_{1,i,m}, e_{3,m,n}) \vee (e_{2,j,m}, e_{4,m,k}) = \\ &= \left((e_1 e_4)_{i,k} \vee (e_1 e_3)_{i,n} \right) \vee \left((e_2 e_3)_{j,n} \vee (e_2 e_4)_{j,k} \right). \end{aligned}$$

В деяких випадках дослідження зв'язаності графа спрощується при визначені його характеристик.

3.5. Виміри графів

Найпростішим виміром графів є цикломатичне число, яке при відповідній модифікації застосовується в програмній інженерії для ана-

лізу структури програм, кваліфікаційного рівня колективів програмістів тощо [3 – 5].

Визначення 3.11. Найменше число вершин графу G, видалення яких приводе до незв'язаного графу, називається *зв'язністю графу* $\kappa(G)$.

Якщо зв'язаний граф має n вершин і m ребер, тоді

Визначення 3.12. Цикломатичне число $\nu(G)$ зв'язаного неорієнтованого графа G підраховується за формулою $\nu(G) = m - n + 1$ і характеризує вимір зв'язаності графу.

Наприклад, зображені на рис. 3.1 графи мають наступні цикломатисні числа: $\nu(G_a) = 6 - 4 + 1 = 3$, $\nu(G_b) = 8 - 6 + 1 = 3$ і $\nu(G_c) = 4 - 5 + 1 = 0$. Це означає наявність трьох циклів у перших двох графах і відсутність циклів у третьому графі.

Виходячи з того, що орієнтований граф з однією початковою і однією заключною вершинами взагалі утворений декількома альтернативними маршрутами, тобто не сильно зв'язаний, тому вводячи віртуальну дугу з порожньою позначкою o, яка з'єднує заключну і початкові вершини, отримаємо сильно зв'язаний граф і тому маємо можливість підрахувати цикломатичне число і для цього випадку [5] $\nu(G) = m - n + 2$.

На графах, зображені на рис. 3.6 віртуальні дуги зображені пунктиром і їх цикломатичні числа $\nu(G_a) = \nu(G_b) = 5 - 4 + 2 = 3$.

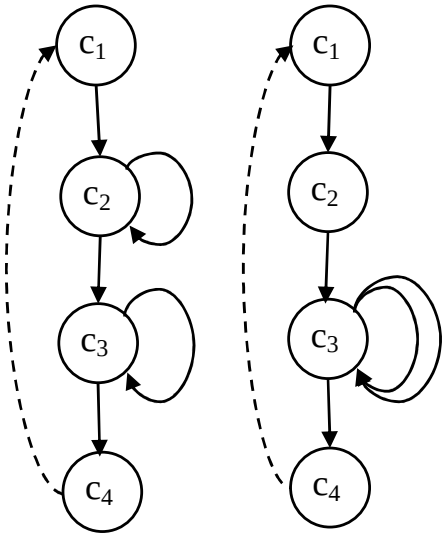


Рис. 3.6. Графи з віртуальними дугами для виміру цикломатичного числа

З наведеного прикладу видно, що цикломатичне число не враховує складності графів. Очевидно, що другий граф за своєю природою складніший першого, хоча у них цикломатичні числа однакові. *Складність* графу можна визначити різним чином. Наприклад, якщо визначити складність графу

$$\varsigma(G) = \max_{i \in I}\{st(c_i)\}, \tag{3.5}$$

тоді для першого графу на рис 3.6 $\varsigma = 4$, для другого – $\varsigma = 6$.

Але складність за виразом (3.5) не враховує будову графу. Виправити це можна за допомогою введення структурних векторів графу [3, 9]. *Перший структурний вектор графу* визначається на степенях вершин графу, тому назвемо його *степеневим структурним вектором* $Q(G)$ графу G. Для побудови вектору Q розіб'ємо вершини графу G по степеням s на класи Q_s. Тоді

$$Q(G) = \left(\#Q_0, \#Q_1, \cdots, \#Q_\varsigma\right), \tag{3.6}$$

де $\#Q_s$ – розмір класу Q_s, індекс ς визначається формулою (3.5).

Зрозуміло, що кількість вершин графу G завжди дорівнює $\sum_{k=0}^{\varsigma} \#Q_k$. Але кількість дуг графу за степеневим вектором не очевидна. Аналіз виразів (3.5) і (3.6) показує, що вимір складності графу завжди можна знайти із структурного вектора (3.6), причому, чим більший розмір вектору, тим складнішим буде граф.

Знайдемо вектори $Q(G_1)$ і $Q(G_2)$ для графів G_1 і G_2 рис. 3.6 (віртуальні дуги не належать графам). Для першого графу: $\varsigma(G_1) = 4$, $Q_0 = \varnothing$, $Q_1 = \{c_1, c_4\}$, $Q_2 = \varnothing$, $Q_3 = \varnothing$ і $Q_4 = \{c_2, c_3\}$, тому степеневий структурний вектор $Q(G_1) = (0, 2, 0, 0, 2)$. Аналогічно знаходимо степеневий вектор для другого графу $Q(G_2) = (0, 2, 1, 0, 0, 0, 1)$. Порівнюючи вектори $Q(G_1)$ і $Q(G_2)$ бачимо, що розмір другого вектора більший, тому відповідний граф складніший.

Розглянемо тепер *другий структурний вектор графу* визначений на відстанях між його вершинами. Нехай c_i і c_j довільні зв'язані вершини графу G, тоді між ними за формулою (3.3) існує відношення відстані $\rho(c_i, c_j)$. Можна перевірити, що ρ є відношенням еквівалентності, тому множину вершин графу G за цим відношенням поділено на класи еквівалентності W_s; $s = \rho(c_i, c_j)$. Очевидно, що будь який клас W_s складається з множини пар вершин, відстань між якими дорівнює s; зокрема, клас W_0 складається з пар (c_i, c_i) всіх вершин графу G. Тепер, другий структурний вектор графу визначимо так:

$$W(G) = \left(\#W_0, \#W_1, \cdots \#W_d\right), \tag{3.7}$$

де d – діаметр графу (див. рис. 3.4).

Другий структурний вектор графу за змістом пов'язаний з вимірами відстаней між вершинами цього графу, тому назвемо його *метричним структурним вектором графу*. Очевидно, наявність петель на вершинах графу не впливає на метричний вектор, але наявність циклів і розгалужень змінюють його структуру.

Підрахуємо компоненти векторів $W(G_1)$ і $W(G_2)$ для графів рис. 3.6.

$W_0 = \{(c_1, c_1), (c_2, c_2), (c_3, c_3), (c_4, c_4)\}$,

$W_1 = \{(c_1, c_2), (c_2, c_3), (c_3, c_4)\}$,

$W_2 = \{(c_1, c_3), (c_2, c_4)\}$ і

$W_3 = \{(c_1, c_4)\}$.

Зрозуміло, що ці компоненти однакові для обох метричних векторів, з відси маємо: $W(G_1) = W(G_2) = (4, 3, 2, 1)$. При наявності циклів (врахувати віртуальні дуги) в графах G_1 і G_2 рис. 3.6 компоненти і структура їх метричних векторів змінюється $W(G_1) = W(G_2) = (4, 4, 2)$.

Зауважимо, що розглянуті виміри графу: цикломатичне число, діаметр, складність, структурні вектори є інваріантними до орієнтації графу.

3.6. Завдання і вправи

1. На вершинах $C = \{1, 2, 3, 4, 5\}$ $C = \{1, 2, 3, 4, 5\}$ побудувати орграф, якщо множини D D визначаються відношеннями:

а) $\rho_1 = \begin{pmatrix} 2 & 1 \\ 2 & 3 \end{pmatrix}$, $\rho_2 = \begin{pmatrix} 3 & 1 \end{pmatrix}$, $\rho_3 = \begin{pmatrix} 4 & 1 \\ 4 & 3 \end{pmatrix}$, $\rho_4 = \begin{pmatrix} 5 & 1 \\ 5 & 2 \\ 5 & 3 \\ 5 & 4 \end{pmatrix}$;

б) $\rho_1 = \begin{pmatrix} 2 & 2 \\ 2 & 3 \end{pmatrix}$, $\rho_2 = \begin{pmatrix} 1 & 4 \\ 1 & 1 \end{pmatrix}$, $\rho_3 = \begin{pmatrix} 4 & 1 \\ 4 & 3 \end{pmatrix}$, $\rho_4 = \begin{pmatrix} 5 & 1 \\ 5 & 2 \\ 5 & 3 \\ 5 & 5 \end{pmatrix}$;

$\rho_1 = \begin{pmatrix} 1 & 1 \\ 1 & 3 \end{pmatrix}$, $\rho_2 = \begin{pmatrix} 3 & 1 \end{pmatrix}$, $\rho_3 = \begin{pmatrix} 4 & 1 \\ 4 & 3 \end{pmatrix}$, $\rho_4 = \begin{pmatrix} 5 & 3 \\ 5 & 1 \end{pmatrix}$, $\rho_5 = \begin{pmatrix} 2 & 2 \end{pmatrix}$;

г) $\rho_1 = \begin{pmatrix} 1 & 2 \\ 1 & 3 \end{pmatrix}$, $\rho_2 = \begin{pmatrix} 3 & 5 \end{pmatrix}$, $\rho_3 = \begin{pmatrix} 4 & 1 \\ 4 & 4 \end{pmatrix}$, $\rho_4 = \begin{pmatrix} 5 & 1 \\ 5 & 2 \\ 5 & 5 \\ 5 & 4 \end{pmatrix}$.

2. Для графів попередньої вправи з'ясувати степені вершин. Визначити які з них ізольовані.
3. Для графів вправи 1 виділити шляхи та визначити їх довжину. Які з них утворюють маршрути і цикли?
4. Які із графів вправи 1 утворюють дерева?
5. На прикладах вправи 1 пояснити зв'язаність графів.
6. Для прикладів вправи 1 побудувати матриці суміжностей.
7. Якщо вважати графи вправи 1 навантаженими, тоді спростити графи (якщо це можливо).
8. Для графів вправи 1 побудувати ізоморфні графи.
9. За допомогою бінарного алфавіту побудувати кодові дерева для кодування у восьмирічній та шістнадцятирічній системах числення.
10. Відтворити дерево для розпізнання задуманого партнером числа від одиниці до n, $n = 8$, $n = 15$, $n = 33$. Якою є довжини маршрутів цих дерев?
11. Побудувати дерево для знаходження фальшивої монети серед купи із m монет за допомогою ваг, якщо відомо, що фальшива монета важча за звичайну монету. Чому дорівнює степінь вершин цього дерева?
12. Для графів вправи 1 знайти структурні вектори.

13. Довести, що вимір відстані між вершинами графу задає відношення еквівалентності.

3.7. Підсумковий коментар

Розглянуті у розділі елементи теорії графів необхідні студентам, які навчаються за напрямком програмної інженерії по-перше, для засвоєння навичок організації і обробки графових структурами даних [2, 13], по-друге для здобуття знань з аналізу і перетворення програм [9 – 12]. Теорія графів незавершена і продовжує розвиватися як класична конструктивна наука, так і прикладна наука програмної інженерії. Теорія графів застосовується у нечітких структурах, її поширюють на некласичні множинні об'єкти, зокрема, на мультографи і мультомережі [1].

Література до розділу 3

1. *Андрющенко В.А., Ильман В.М., Шинкаренко В. И.* Конструктивное представление мультиграфов. // Вісник Дніпропетровського національного університету імені академіка В. Лазаряна. – 2011.– Вип. 36. – С. 161-167.
2. *Ахо А.В., Хопкрофт Д., Ульман Д.Д.* Структуры данных и алгоритмы: Пер. с англ. – М.: «Издательский дом Вильямс», 2003. – 384 с.
3. *Босов А.А., Ильман В.М.* Структурная сложность систем // Вісник. ДНУЗТ Вип. 40. Дніпр. 2012. – С. 173-179.
4. *Горбатов В. А.* Основы дискретной математики. М.: Высш. шк., 1986. – 311 с.
5. *Евстигнеев В. А., Кожевникова Г. П.* Топологическая сложность программ: – М.: Препринт / АН СССР. Ин-т точной механики и вычисл. техники. Новосиб. фил. ; № 23, 1989. – 30 с..
6. *Ільман В. М., Скалозуб В. В., Шинкаренко В. І.* Утворюючі системи графів // Вісник Дніпропетровського національного університету імені академіка В. Лазаряна. –2007 – вип. 17. – С.127-133.
7. *Ільман В. М., Скалозуб В. В., Шинкаренко В. І.* Відтворення графів за технологічними шляхами // Вісник Дніпропетровського національ-

ного університету імені академіка В. Лазаряна. –2007 – вип. 18. – С. 85-94.
8. *Ільман В. М., Скалозуб В. В., Шинкаренко В. І.* Формальні структури та їх застосування: монографія. – Д.: Вид-во Дніпроп. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. – 296 с.
9. *Касти Дж.* Большие системы. Связность, сложность и катастрофы: Пер. с англ. – М.: Мир, 1982. – 216 с.
10. *Касьянов В.Н., Евстигнеев В.А.* Графы в программировании: обработка, визуализация и применение. – СПб. БХВ-Петербург, 2003. – 1104 с.
11. *Нечипоренко В. И.* Структурный анализ систем (эффективность и надежность): монография. – М.: Сов. радио, 1977. – 216 с.
12. *Свами М., Тхуласираман К.* Графы, сети и алгоритмы. – М.: Мир, − 1984 – 380 с.
13. *Топп У., Форд У.* Структуры данных в C++: Пер. с англ. – М.: ЗАО «Издательство БИНОМ», 1999. – 816 с.

Розділ 4.
Абстрактні моделі і структури в теорії програмування

У цьому розділі розглядаються важливі для теоретичного і практичного програмування абстрактні математичні об’єкти такі як автомати, алгебраїчні та формально конструктивні структури. Загальні поняття автоматів, виникли і сформовані у середині попереднього сторіччя [2, 4, 16, 17]. Алгебраїчні структури як математичні об’єкти суттєво почали розглядатися також у середині попереднього сторіччя Біргофом [3], Бурбаки [5], Мальцевим [11] та іншими дослідниками. Моделі автоматів широко застосовуються в теорії формальних мов, теорії алгоритмів, базах знань та інших розділах програмної інженерії. Алгебраїчні структури застосовуються в першу чергу для аналізу операцій та алгоритмів, дослідження їх властивостей, виконання різних перетворень над об’єктами програм [1, 2, 10, 12, 13, 18] тощо.

4.1. Мета розділу

Теорія програмування не завершена наука і має розмиті межі. Однією із цілей якої є розвиток математичного апарату, орієнтованого на теоретичне практичне програмування. Ці дослідження спираються на теорії абстрактних автоматів, алгебр, структур і інше, викладення елементів яких наведено у розділі.

Метою розділу є наведення основних відомостей про різні абстрактні автомати, їх способи представлення і задавання, мови автоматів та інше; різновиди абстрактних алгебр і формальних конструктивних структур, якими необхідно володіти студентам для засвоєння навичок моделювання алгоритмічних структур, розробки алгоритмічних програм і штучних систем.

4.2. Абстрактні автомати

У своїй діяльності людина стикається з автоматами: торговельними автоматами, виробничими керуючими автоматами тощо. Різновид автоматів досить великий і їх конструктивне представлення різноманітне. Але характерним для них є те, що вони функціонують в часі, мають входи, виходи і знаходяться в певних станах. Наприклад, інформаційний автомат приймає повідомлення (стан прийняття інформації), переробляє інформацію (стан кодування), передає сигнал (стан відправлення сигналу) та інші. Нас будуть цікавити *абстрактні автомати*, як моделі реальних пристроїв, процесів та інше. Абстрактні автомати застосовуються в різних системах управління, штучного інтелекту. Різні види абстрактних автоматів використовують для обробки слів (ланцюжків) формальних мов (*словарні автомати),* для моделювання пам'яті обчислюваних машин (*автомати над пам'яттю),* для моделювання множин мереж (*мереживі автомати)* і інші.

Змістовно абстрактний *автомат* можна представити як абстрактний пристрій призначений для перетворення вхідних сигналів у вихідні, тому він має вхідний та вихідний канали зв'язку і знаходиться у кожний дискретний момент часу t_k в одному із станів $z_i(t_k) \in Z$. Причому у ці моменти часу через вхідний канал зв'язку до пристрою поступають сигнали з повідомленнями у вигляді слів (ланцюжків), побудованих на символах (об'єктах) деякого вхідного скінченого об'єктного алфавіту A. В ті ж моменти часу через вихідний канал зв'язку пристрій видає сигнал повідомлення, яке складається з елементів вихідного об'єктного алфавіту B, так представлений автомат нагадує пристрій кодування інформації – з одного боку, а з іншого – формальну систему.

Абстрактні автомати поділяють на *скінченні* та *нескінченні*, якщо автомат може знаходиться тільки у скінченій сукупності станів, тому його називають скінченим, у протилежному випадку нескінченим.

Приклад 4.1. Побудуємо макромодель автомату $\mathbb{A}$, яка реалізує процес функціонування СМО з прикладу 1.2.

▷ Пристрій, який реалізує СМО складається з множини терміналів T, (їх два), котрі зв'язані каналами з автоматом A (рис. 4.1).

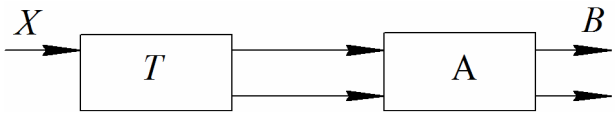


Рис. 4.1. Технічний пристрій СМО

Залежність виходів автомату (закритий або відкритий канал автомату) від входів задається функцією істинності (див. табл. 1.3). Отже, маємо чотири вхідних комбінацій-кортежів автомату, які позначимо як $a=(0,0)$, $b=(0,1)$, $c=(1,0)$, $d=(1,1)$ і вони утворюють множину входів – алфавіт $A=\{a,b,c,d\}$.

Автомат має зачинений або відкритий канали зв'язку, тому з формального боку він може знаходитися у восьми станах z, пов'язаних з входами і виходами

$$z_1=z(a,1),\ z_2=z(b,1),\ z_3=z(c,1),\ z_4=z(d,1);$$

$$z_5=z(a,0),\ z_6=z(b,0),\ z_7=z(c,0),\ z_8=z(d,0).$$

За значеннями функції істинності, (табл. 1.3) наведені стани можуть бути помилковими, тому автомат повинен видавати повідомлення про хибність стану (значення – 0). Хибність стану автомату змінюється в наступні такти часу у разі подання на термінали тільки позитивного входу (комбінація d). Таким чином запирання і відкриття каналу (0 або 1) характеризується значенням логічної функції (0 – хибно, 1 – істинно) і множина виходів $B=\{p,q,w,r\}$ автомату визначається відповідними парами $(0,0)=p$, $(1,0)=q$, $(0,1)=w$ і $(1,1)=r$. Так як автомат функціонує у часі t, тоді його стани змінюються через постійні часові кванти Δt за правилом

$$z_i(t)\to(z_j(t+\Delta t),b_k), \qquad (4.1)$$

у якому права частина заміщення визначена на декартовому добутку $Z\times B$.

Схема функціонування автомату буде складатися з набору правил (4.1), яку зручно представити табл. 4.1.

Зауважимо, що у табл. 4.1 замість кортежів (z_i,b_j) використана спрощена форма представлення z_i/b_j.

Таблиця 4.1

Таблиця переходів автомату

	a	b	c	d
z_1	z_1/q	z_2/q	z_1/r	z_8/p
z_2	z_5/p	z_2/q	z_3/q	z_2/r
z_3	z_3/r	z_6/p	z_3/q	z_4/q
z_4	z_1/q	z_4/r	z_7/p	z_4/q
z_5	z_5/p	z_2/q	z_5/w	z_8/p
z_6	z_5/p	z_6/p	z_3/q	z_6/w
z_7	z_7/w	z_6/p	z_7/p	z_4/q
z_8	z_1/q	z_8/w	z_7/p	z_8/p

Зауважимо, що у табл. 4.1 замість кортежів (z_i, b_j) використана спрощена форма представлення z_i/b_j.

Відслідковувати роботу автомату також можна за його графом, котрий зображено на рис. 4.2.

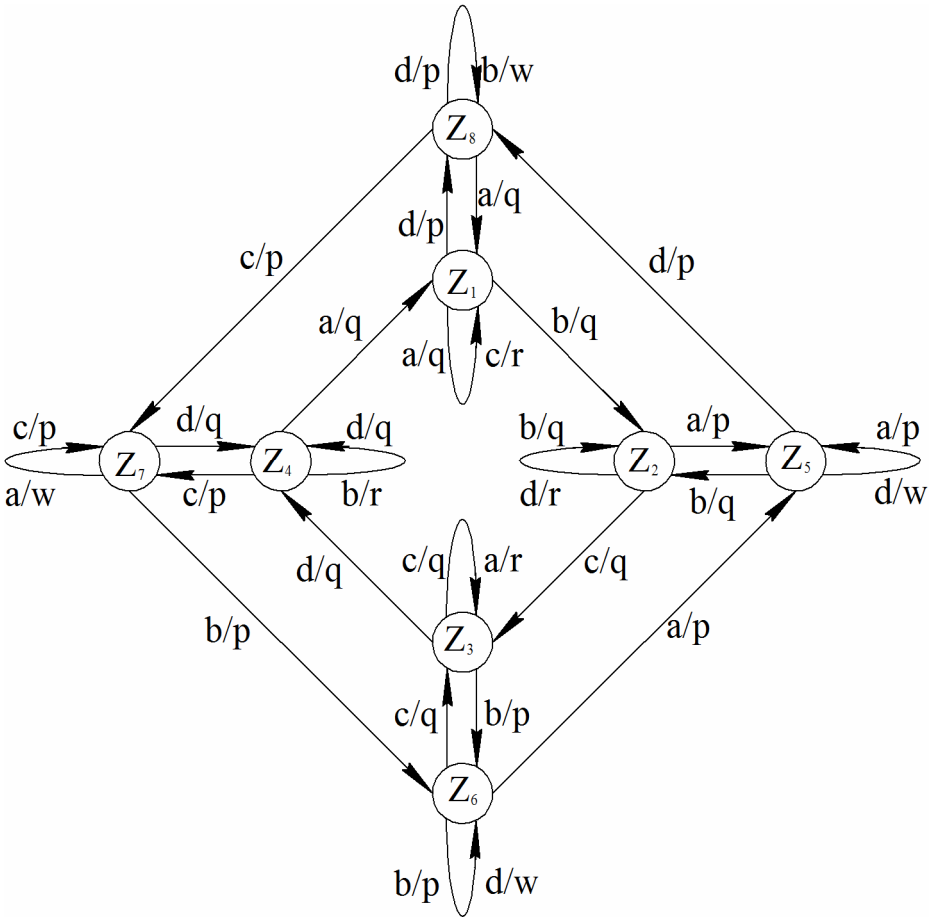


Рис. 4.2. Граф переходів автомату узгодженого з даними табл. 4.1

Напрямок дуг графу визначається правилом обробки табл. 4.1 «від вершини першої колонки вибраного рядка до вершини, вказаній у відповідній клітині цього рядка». ◁

Звернемо увагу на те, що розглянутий автомат $\mathbb{A}$ схожий з технічним автоматом обертання вала, який розглянуто в роботі [4]

4.2.1. Визначення та способи визначення автоматів

Існує декілька підходів до визначення автоматів. Якщо розглядається зовнішня поведінка пристрою, тобто *функціонування* пристрою по відношенню до зовнішнього середовища, тоді застосовують підхід *чорного ящика* так званий *макропідхід,* якщо розглядається його поведінка в залежності від станів елементів пристрою, їх зв'язаності та інше, тоді застосовується підхід типу *білого ящика* або *мікропідхід.*

Макропідхід застосовано у прикладі 4.1. При макропідході автомат визначається так [4, 17]:

Визначення 4.1. Скінчений автомат це формальна упорядкована п'ятірка

$$\mathbb{A} = \langle A, Z, B, \varphi, \psi \rangle, \tag{4.2}$$

в якій A – *вхідний* скінчений алфавіт, Z – скінчена *множина станів*, B – *вихідний* скінчений алфавіт, φ – *функція переходів* визначена як $\varphi: A \times Z \to Z$ і ψ – *функція виходів*, яка відображає декартовий добуток $A \times Z$ в B.

За підходом чорного ящика абстрактний автомат (4.2) можна зобразити, як на рис. 4.3.

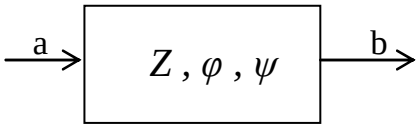


Рис. 4.3. Представлення автомату «чорним ящиком»

Представлення автомату у вигляді зображеного на рис. 4.3 задає його *агрегативну модель* з довільним входом $a \in A$ та довільним виходом $b \in B$, яка знаходиться у стані $z(t) \in Z$ під впливом *оператора станів* φ і *оператора виходів* ψ.

Визначені за виразом (4.2) автомати також звуть автоматами *Мілі* [2].

Розглянемо питання функціонування автомату $\mathbb{A}$ за введеним визначенням 4.1.

Нехай автомат $\mathbb{A}$ знаходиться в момент часу t_0 у деякому початковому стані $z(t_0) \in Z$ та готовий до прийняття повідомлення у вигляді ланцюжка $l_A = a_1 a_2 \ldots a_n$, визначеного на алфавітові A, тобто $l_A \in \mathbb{F}(A)$ і його вхідний канал приймає посимвольно цей ланцюжок відповідно в моменти часу $t_1 > t_0$, $t_2, \ldots, t_n$ як $a_1(t_1), a_2(t_2), ..., a_n(t_n)$. Тоді функціонування автомату визначене, якщо визначені його стани і виходи в кванти часу t_i, тобто за виразами (4.1) і (4.2) маємо

$$\begin{aligned} z(t_i) &= \varphi(a_{i-1}(t_{i-1}),\, z(t_{i-1})) \in Z, \\ b_i(t_i) &= \psi(a_i(t_i),\, z(t_i)) \in B. \end{aligned} \tag{4.3}$$

Вираз (4.3) означає, що стан $z(t_i)$ автомату залежить від попередніх станів і об'єктів вхідного алфавіту, які прийняті автоматом до моменту часу t_i. Через це вихідні об'єкти автомату b_i також залежать від попередньо прийнятих об'єктів вхідного алфавіту. Отже вхідному ланцюжкові l_A автоматом $\mathbb{A}$ ставиться у відповідність вихідний ланцюжок l_B, побудований на елементах алфавіту B за відображенням (4.3). Множина всіх вихідних ланцюжків автомату $\mathbb{A}$, отриманих за допомогою рекурсивних правил (4.3) з початкового стану z_0, називається *формальною мовою автомату* або просто *мовою автомату* (*мовою проведену автоматом*) і позначається як $L(\mathbb{A})$. Очевидно, що мова автомату є підмножиною вільної мови, тобто $L(\mathbb{A}) \subset \mathbb{F}(B)$, причому може статися, що мова автомату буде порожньою $L(\mathbb{A}) = \varnothing = \{o\}$; нагадуємо, що порожній елемент $o \in A$ і $o \in B$.

Введене визначення автомату є досить загальним, але на практиці виникають часткові випадки автомату $\mathbb{A}$, тому розглянемо деякі з них.

Автомати без пам'яті. Так називають автомати які визначаються трійкою $\langle A, B, \psi \rangle$. Залежності (4.3) в цьому випадку вироджуються в $b_i = \psi(a_i)$, тобто вихідний символ на i-ому моменті часу залежить тільки від вхідного символа автомату в цей же момент часу і не залежить від інших попередніх вхідних символів. Таким чином, кожний автомат відтворює по символьний перевод вхідних символів у вихідні об'єкти. Прикладом такого автомату є кодування символів $n-$ нарної системи числення кодовими комбінаціями визначеними на

деякому алфавітові, наприклад, восьмирічної системи числення $(0,1,...,7)$ – двійковими кодовими комбінаціями $(000,001,...,111)$ (див. рис. 2.1).

Автономні автомати. Це такі автомати які представляються четвіркою $\langle Z, B, \varphi, \psi \rangle$. Характерним для них є те, що відображення

$$\begin{aligned} z(t_i) &= \varphi(z(t_{i-1})), \\ b_{i-1} &= \psi(z(t_{i-1})); \end{aligned} \tag{4.4}$$

при заданому початковому стані автомату, однозначно відтворює вихідний ланцюжок l_B і ланцюжок станів l_Z. Такі автомати, наприклад, реалізують алгоритмічну програму без вхідних параметрів.

Автомати без виходу задаються упорядкованою трійкою $\langle A, Z, \varphi \rangle$. В співвідношеннях (4.2) для цього випадку зостається лише перше, тобто

$$z(t_i) = \varphi(a_{i-1}, z(t_{i-1})) \tag{4.5}$$

Зрозуміло, що автомат без виходу не утворює ніякої мови, але його поведінку можна відслідковувати за його функціонуванням. Наприклад, якщо автомат обробляє вхідний ланцюжок l_A (перевіряє його належність до певної мови $L(A)$) починаючи з початкового стану $z(t_0)$ і при досягненні деякого заключного стану $z(t_k)$ приймає значення *true* (істини), то вважається, що $l_A \in L(A)$, у протилежному випадку, тобто коли $z(t_k) = \mathit{false}$ (хибно) приймається, що $l_A \notin L(A)$. За таким принципом реалізується одна з функцій автомата–транслятора алгоритмічних програм.

Частковим випадком автоматів без виходу є *недетерміновані автомати* які задаються упорядкованою п'ятіркою $\mathbb{A} = \langle A, Z, Z_1, Z_2, \varphi \rangle$. Тут $Z_1 \subset Z$ множина початкових станів автомату – і $Z_2 \subset Z$ – множина заключних станів, причому, як правило, вважається, що їх перетин порожній.

Автомати Мура також як і автомати Мілі визначаються п'ятіркою (4.2), але вихідна функція задається як $\psi : Z \to B$, тут вихідна функція явно не залежить від символів вхідного алфавіту. Будь який автомат Мура є автоматом Мілі тому, що для автомату Мура функцію виходу можна задати на декартовому добутку $\varnothing \times Z$, де $\varnothing = \{o\} \subset A$.

При макропідході – визначення автомату більш різноманітне воно суттєво залежить від способу моделювання складових елементів автомату. Узагальнене визначення для цього підходу може бути, наприклад, таким: описати елементи s_i, з яких складається внутрішнє середовище автомату і схему з'єднання цих елементів. Цей опис автомату можна виконати моделюючи елементи s_i різними способами, наприклад, автоматами Мура. Упорядковуючи агрегати (автомати) по між елементним зв'язкам отримаємо мережеву схему автомату. Наприклад, абстрактна дискретна модель ЕОМ [21] має схему, яка складається з *керуючого* $\mathbb{A}$ і *операційного* $\mathbb{B}$ автоматів. Метою керуючого автомату $\mathbb{A}$ є відтворення вихідних мікро операцій y, які сприймаються операційним автоматом $\mathbb{B}$. В свою чергу операційний автомат породжує сигнали x повідомлень у вигляді кортежів деяких елементарних логічних умов $(\alpha_1, \alpha_2, ..., \alpha_k)$.Структурна схема абстрактної ЕОМ зображена на рис. 4.4.

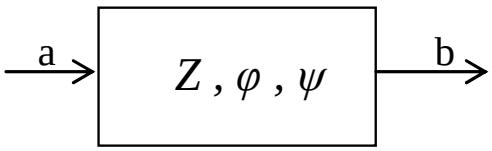


Рис. 4.4. Абстрактна ЕОМ

Керуючий автомат $\mathbb{A}$, як правило представляється скінченим автоматом Мілі, а операційний $\mathbb{B}$ запобігаючи можливостям виникнення зацикленостей представляється нескінченим автоматом Мура стани якого інтерпретуються змістом досить великої кількості регістрів.

Іншим прикладом таких структурних автоматів є так звані логічні схеми, які застосовуються в системах штучного інтелекту.

Розглянемо задавання деяких найбільш поширених можливостей автоматів Мілі. Під висловом: «Задати автомат» розуміється сформувати опис його функціонування (поведінки в часі), тобто подати представлення функцій переходу та виходу в тій чи іншій формі обумовленій предметною областю, на якій розглядається автомат.

Скінчений автомат за представленням (4.2) є формальним конструктивним об'єктом визначеним на вхідному алфавітові та на множині станів, він може бути представлений в площинній *табличній формі* у вигляді двох таблиць відношень: *таблиці переходів* T_φ та *таблиці виходів* T_ψ. Рядки таблиць позначаються символами вхідного алфавіту A, а стовбці – елементами множини станів z і у відповідних клітинах за фо-

рмулами (4.3) записуються елементи множини Z та символи вихідного алфавіту B. Наприклад, якщо $A=\{a,b,c\}$, $B=\{0.1\}$ і $Z=\{\alpha,\beta,\delta\}$, тоді таблиці відношень T_φ та T_ψ можна представити так:

$$T_\varphi = \begin{array}{|c|c|c|c|}\hline a_i \setminus z_i & \alpha & \beta & \delta \\ \hline a & \beta & \alpha & \alpha \\ \hline b & \alpha & \alpha & \beta \\ \hline c & \alpha & \beta & \alpha \\ \hline \end{array}, \qquad T_\psi = \begin{array}{|c|c|c|c|}\hline a_i \setminus z_i & \alpha & \beta & \delta \\ \hline a & 0 & 1 & 1 \\ \hline b & 1 & 0 & 1 \\ \hline c & 0 & 1 & 0 \\ \hline \end{array}.$$

Інколи, позначки та стовпців двох таблиць T_φ та T_ψ, автомат представляють однією таблицею T, в якій у клітинах записують послідовно значення функції φ та ψ. Так для попереднього випадку дві таблиці можна записати як одну

$$T = \begin{array}{|c|c|c|c|}\hline a_i \setminus z_i & \alpha & \beta & \delta \\ \hline a & \beta,0 & \alpha,1 & \alpha,1 \\ \hline b & \alpha,1 & \alpha,0 & \beta,1 \\ \hline c & \alpha,0 & \beta,1 & \alpha,0 \\ \hline \end{array}. \qquad (4.6)$$

Таблиці автоматів, в залежності від їх видів мають специфічну будову. Для автомату Мура таблиця виходів T_ψ має однакові значення у стовпцях для певних елементів множини Z. Автомат без пам'яті описується тільки однією таблицею T_φ, наприклад:

$$\begin{array}{|c|c|c|c|}\hline a_i \setminus z_i & \alpha & \beta & \delta \\ \hline a & \beta & & \alpha \\ \hline b & & \alpha & \beta \\ \hline c & \alpha & \beta & \\ \hline \end{array},$$

порожні клітини означають, що на відповідних елементах функція φ не задана.

При досліджені, представлених у табличній формі автоматів можна користуватися прийомами досліджень відношень, як у розділі 2.

Таблична форма представлення автомату дозволяє розглядати її клітини, як результати операції заміщення в формальній системі (див. розділ 1) і представляти ці таблиці у вигляді *правил формальної сис-*

теми автомату. Для прикладу, за вище наведеною таблицею (4.6) система правил має вигляд:

$$\begin{aligned} &p_1 : a\alpha \to \beta 0,\ \ p_2 : a\beta \to \alpha 1,\ \ p_3 : a\delta \to \alpha 1; \\ &p_4 : b\alpha \to \alpha 1,\ \ p_5 : b\beta \to \alpha 0,\ \ p_6 : b\delta \to \beta 1; \\ &p_7 : c\alpha \to \alpha 0,\ \ p_8 : c\beta \to \beta 1,\ \ p_9 : c\delta \to \alpha 0. \end{aligned} \qquad (4.7)$$

Припустимо, що α початковий стан, а β заключний стан автомату і нехай в моменти часу t_0, t_1, t_2 на вхід автомату подаються символи ланцюжка acc, тоді після переробки ланцюжка автоматом за правилами (4.7) в момент часу t_3 отримаємо на виході код цього ланцюжка 011.

Автомати також зручно представляти в наглядній *графовій формі* у вигляді площинного орієнтованого мультографа, за вершини якого приймають стани автомату $z_i \in Z$, а дуги навантажують елементами декартового добутку $A \times B$. Так за першим правилом автомату (4.7) маємо дугу визначену кортежем (α, β) і навантажену упорядкованою парою $(a, 0)$ за другим правилом отримаємо дугу (β, α) навантажену парою $(a, 1)$ і так далі. Побудований за правилами (4.7) граф автомату зображено на рис. 4.5.

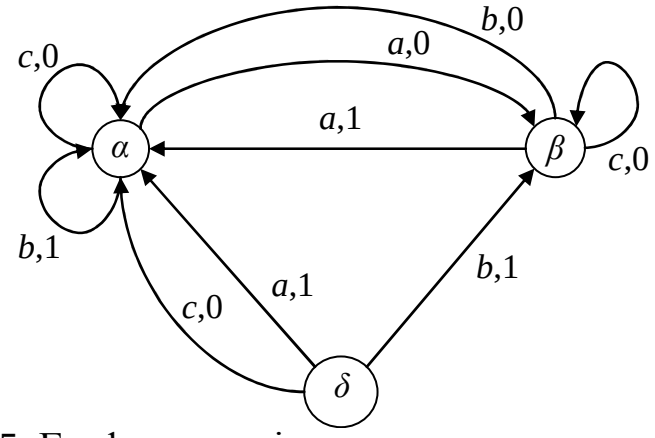


Рис. 4.5. Граф переходів автомату за правилами (4.7)

В залежності від типу автомату його графове представлення може мати ті чи інші особливості. Так автономний автомат описується графом, з кожної вершини якого виходить тільки дуга помічена символом вихідного алфавіту; для автомату Мура характерним є те, що вхідні дуги в вершину z_i мають однакові вихідні позначки, як зображено на рис. 4.5 і тому ці позначки можна віднести до позначки відповідної вершини.

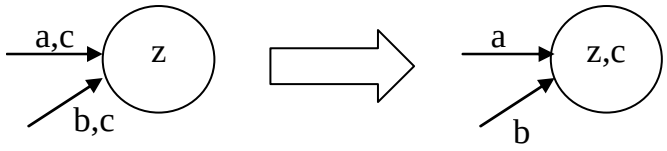


Рис. 4.6. Фрагмент представлення графу переходів автомату Мура

Для недетермінованих автоматів їх графи мають вершинами стани автоматів, а дуги навантажуються символами вхідних алфавітів. Причому недермінованість автомату характеризується тим, що позначки суміжних дуг можуть бути однаковими.

Як відомо, граматичні правила алгоритмічних мов програмування інколи зображають *синтаксичними діаграмами*. Наприклад, правило побудови ідентифікатора (ім'я змінної, ім'я файлу тощо) зображується такими синтаксичними діаграмами (рис. 4.7).

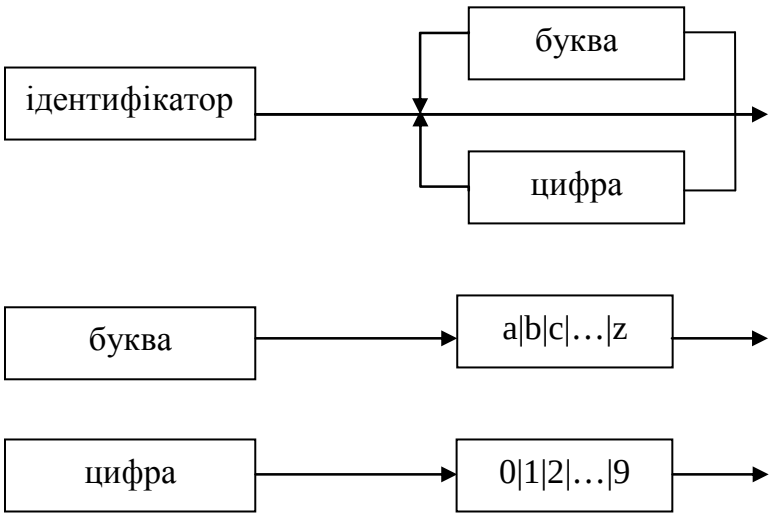


Рис. 4.7. Синтаксичні діаграми ідентифікатора

З синтаксичних діаграм зображених на рис. 4.7 видно, що ідентифікатор є ланцюжок (можливо нескінчений), який складається з букв (в нашому випадку малих латинських) і цифр, причому початковим символом ланцюжка є буква. Якщо тепер ввести позначки: z_0 – початковий стан автомату, z_i – заключний стан (ідентифікатор) автомату, навантажити довільними буквами з позначкою «б» і довільними цифрами з позначкою дуги «ц», тоді граф автомату прийме вигляд зображений на рис. 4.8.

Формальна мова, яка проводиться цим автоматом може бути представлена так: $L(\mathbb{A}) = \{б \mid ((б)^n \mid (ц)^n)^n;\ n = 0,1,2,...\}$.

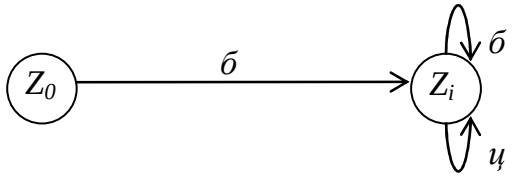


Рис. 4.8. Граф переходів автомату, який проводе ідентифікатори

Добре розвинута теорія графів дозволяє перенести визначення і поняття з графів на автомати. Наприклад, автомат $\mathbb{A}_1$ є підавтоматом автомату $\mathbb{A}$, $\mathbb{A}_1 \subset \mathbb{A}$, якщо між їх відповідними графами існує включення $G_1 \subset G$; автомати $\mathbb{A}_1$ і $\mathbb{A}_2$ ізоморфні, якщо їх графи G_1 і G_2 ізоморфні (див. розділ 3).

Існують інші методики опису поведінки автоматів, наприклад, матричне представлення, представлення автоматів за допомогою логічної мови одномісних предикатів [4, 17, 20] тощо. Але ми будемо надавати перевагу наглядній графічній формі представлення автоматів.

4.2.2. Недерміновані автомати

Розглянемо деякі елементи досліджень і властивостей не детермінованих автоматів. Нагадаємо, що не детермінований автомат визначається як

$$\mathbb{A} = \langle A, Z, Z_0, Z_1, \varphi \rangle, \tag{4.8}$$

або його можна задати звичайним орієнтованим мультографом G з множиною початкових станів $Z_0 \subset Z$ та заключних – $Z_1 \subset Z$. Через це кожна пара суміжних (z_i, z_j) вершин графу G має різні позначки символів (дуги з однаковими позначками інформативно нічого нового не несуть) з деякого підалфавіту $A_k \subseteq A$, що дає можливість підграфові на кортежі (z_i, z_j) поставити в однозначну відповідність множину $A_{k,i,j}$. Серед підмножин алфавіту A можна виділити *суміжні підмножини* $A_{k,i,j}, A_{m,j,n} \subseteq A$, тобто такі, які визначені на спільних станах автомату. Зрозуміло, що послідовність попарно суміжних підмножин алфавіту A $\left(A_{1,1,2}, A_{2,2,3}, ..., A_{k,k,m}\right) = N_{1,m}$ утворює *напрямок поведінки* автомату від стану z_1 до стану z_m. Якщо множини $A_{1,j,n}$

одноелементні, то маємо звичайний шлях $P_{1,m}$ графу G або *шлях поведінки* автомату. причому має місце включення $P_{1,m} \subseteq N_{1,m}$, за яким кожен елемент шляху належить відповідній множині напрямку. Так, наприклад, напрямок на вершинах (1,2,3,4) $N = \left(A_{1,1,2}, A_{2,2,3}, A_{3,3,4}, A_{4,4,2} \right)$, де $A_1 = \{a, c\}$, $A_2 = \{a, b, c\}$, $A_3 = \{b\}$ і $A_4 = \{b, d\}$ – підмножини множини $A = \{a, b, c, d\}$; включає шляхи $P_1 = (a_{1,2}, c_{2,3}, b_{3,4}, b_{4,2})$ і $P_2 = (c_{1,2}, b_{2,3}, b_{3,4}, d_{4,2})$, котрі є частковим випадком напрямків поведінки автомату і їм можна передати альтернативи напрямків поведінки виключивши паралельні дуги за правилами виключень на графах (див. розділ 3). Тобто замість напрямку поведінки автомату можна розглядати *альтернативний шлях поведінки автомату*

$$P = \left((a \vee b)_{1,2}, (a \vee b \vee c)_{2,3}, b_{3,4}, (b \vee d)_{4,2} \right). \qquad (4.9)$$

В подальшому напрямки поведінки автомату будемо представляти виразом альтернативного шляху (4.9). Розглянемо деякі властивості альтернативних шляхів поведінки автоматів.

Шлях P^* поведінки автомату назвемо *простим* якщо він відтворює поведінку автомату між суміжними станами графу G. Очевидно, простий шлях може відтворювати поведінку автомату на одному стані за петлею графа. Два прості шляхи P_1^* і P_2^* назвемо *суміжними шляхами*, якщо вони відтворюють поведінку автомату з суміжними підмножинами $A_{k,i,j}, A_{m,j,n} \subseteq A$. Шлях поведінки автомату є *складним шляхом*, якщо цей шлях складається з простих шляхів. Шлях P^* можна отримати за допомогою об'єднання суміжних шляхів, наприклад, шлях (4.9) утворений за допомогою операції об'єднання простих суміжних шляхів: $\left((a \vee b)_{1,2} \right)$, $\left((a \vee b \vee c)_{2,3} \right)$, $\left(b_{3,4} \right)$ і $\left((b \vee d)_{4,2} \right)$. Шлях поведінки P^* автомату є *тупиковим шляхом*, якщо він завершується в тупиковій вершині графу або в заключному стані автомату. *Повним шляхом поведінки автомату* є тупиковий шлях, який починаються в початковому стані автомату.

Можна досліджувати будову автоматів за їх шляхами та під шляхами аналізуючи структури графів автоматів, структурні вектори тощо.

4.2.3. Мови автоматів і методи їх побудови

Позначки альтернативних шляхів автомату утворюють *мовні конструкції* на його вхідному алфавітові. Так альтернативний шлях (4.9) утворює мовну конструкцію

$$L_i = \left\{ (a \vee b)((a \vee b \vee c)b(b \vee d))^n (a \vee b \vee c)b;\ n \in \mathbb{N}_0 \right\}.$$

Очевидно, мовні конструкції L_i, утворені повними шляхами поведінки автомату, є підмножинами мови автомату $L(\mathbb{A})$.

В загалі мовні конструкції можуть належати чи не належати до мови автомату.

Розв'язання задачі належності мовної конструкції L_i можна за допомогою *прийому розшифровки* мови автомату. Прийом розшифровки мови автомату полягає у наступному:

– для заданого автомату (4.8) будуємо його мультограф G;

– для кожної початкової вершини $z_{0i} \in Z_0$ графу G будуємо всі можливі повні альтернативні шляхи поведінки автомату;

– за всіма повними альтернативними шляхами знаходимо мовні конструкції L_j;

– об'єднуючи всі мовні конструкції $\bigcup_j L_j$ отримаємо мову автомату.

Також за допомогою прийому розшифровки можна дати відповіді на питання належність мовної конструкції до мови $L(\mathbb{A})$, чи автоматна мова є порожньою тощо.

Запропонуємо інший більш зручний прийом побудови мови автомату за так званим *функціонально еквівалентним двополюсником*.

Спочатку введемо необхідне поняття еквівалентності автоматів.

Визначення 4.2. Автомати $\mathbb{A}_1 = \langle A, Z_1, Z_{10}, Z_{11}, \varphi_1 \rangle$ і $\mathbb{A}_2 = \langle A, Z_2, Z_{20}, Z_{21}, \varphi_2 \rangle$ *функціонально еквівалентні*, якщо вони проводять одну і ту ж мову, тобто $L(\mathbb{A}_1) = L(\mathbb{A}_2)$.

За роботою [17] визначимо двополюсник, як не детермінований автомат з одним початковим станом z_0 і одним z_k – заключним станом $\mathbb{A}_d = \langle A, Z, z_0, z_k, \varphi \rangle$, для графу якого виконуються наступні умови:

– з початкового стану z_0 тільки виходять дуги,

– в заключний стан z_k тільки входять дуги.

Тоді у початковому і заключному станах такого автомату петлі і контури на графові відсутні.

Властивості не дермінованих автоматів визначає

Твердження 4.1. Будь який не детермінований автомат зводиться до функціонально еквівалентного йому двополюсника.

▷ Це твердження виникає з того, що для довільного автомату (4.8) можна ввести два додатні стани, один з яких прийняти за початковий z_0 і з'єднати його вихідними дугами (навантаженими порожнім символом o) з усіма початковими станами $z_{i0} \in Z_0$ заданого автомату; другий прийняти за заключний z_k, з'єднавши його порожніми дугами з усіма $z_{j1} \in Z_1$ автомату і переозначивши, якщо потрібно, вершини нового графу переходів, отримаємо функціонально еквівалентний автомат $\mathbb{A}_d$.◁

Твердження 4.2. Будь який двополюсник зводиться до функціонально еквівалентного йому простого шляху.

▷ Це твердження доводиться за допомогою операцій виключення підграфів, розглянутих у розділі 3. ◁

Отже за твердженнями 4.1 і 4.2, мову яку проводить автомат можна отримати наступним шляхом:

– заданий не детермінований автомат звести до двополюсника;
– за допомогою операцій виключення вершин та дуг графу звести двополюсник до простого шляху;
– навантаження простого шляху прийняти за мову автомату.

Розглянуті вище автомати, які проводять ту чи іншу побудовану на вхідному алфавітові мову інколи називають автоматами обробки слів або просто *словарними автоматами*. Виходячи з визначення еквівалентності словарних автоматів можна зробити висновок, що клас функціональної еквівалентності для певної автоматної мови взагалі нескінченний і для заданої мови можливо побудувати скільки завгодно словарних автоматів, які її проводять. У зв'язку з цим виникають різні задачі, наприклад, задача існування в класі еквівалентності автоматів з певними обмеженнями. Розглянемо один з таких автоматів.

Словарний автомат (не детермінований) $\mathbb{A}$ є *детермінованим*, якщо на його графове представлення накладено умови:

1) граф має тільки одну вхідну вершину;
2) кількість вихідних дуг з кожної вершини графу співпадає з розміром вхідного алфавіту автомату;
3) кожна вихідна дуга з будь якої вершині навантажена різними символами вхідного алфавіту;

4) кожна вершина графу досягається з вхідної вершини, тобто між вхідною і будь якою іншою вершиною існує шлях поведінки автомату.

Представимо детермінований автомат як $\mathbb{A}_g = \langle A, Z, z_0, Z_1, \varphi \rangle$. Прикладом такого автомату є автомат з графом на рис. 4.9, де $A = \{a, b\}$.

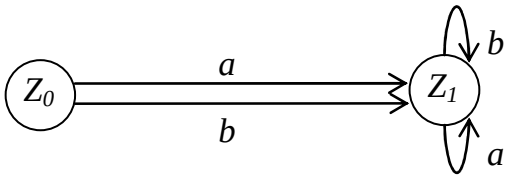


Рис. 4.9. Детермінований словарний автомат

З'ясуємо яким чином пов'язані між собою не детермінованими та детермінованими автоматами. З цього питання в роботі [10, 17] розглянуто теорему про детермінізацію.

Теорема 4.1. Для будь якого не детермінованого автомату завжди можна побудувати йому функціонально еквівалентний детермінований автомат.

Деякі практичні прийоми приведення не детермінованих автоматів до еквівалентних детермінованих будуть розглянуті у розділі формальних граматик.

4.2.4. Автомати над пам'яттю

Автомати над пам'яттю э частковим випадком нескінчених автоматів. Конструктивно автомати над пам'яттю, як правило, пов'язують з носієм інформації біологічних об'єктів, технічних пристроїв і інше. Абстрактно інформаційна модель пам'яті може зображатися у вигляді нескінченних стрічок, поділених на комірки [21]. На носіях інформації в певні моменти часу знаходяться визначенні дані. Доступ до даних відбувається за допомогою дій «записати» та «читати». Тоді, за загальним визначенням автомату, автомат над пам'яттю можна представити у вигляді (4.2). В якому A – множина скінчених вхідних алфавітів, Z – множина довільних перелічених множин станів інформаційного носія, B – множина скінчених вихідних алфавітів, φ – множина функцій, які реалізують дію «записати», ψ – множина функцій, які реалізують дію «читати».

Розглянемо питання функціонування автомату над пам'яттю на прикладі одно стрічкового носія.

Нехай вхідним алфавітом є $A=\{a_1,a_2,\ldots a_n\}$ і $B=A$. Припустимо, що:

– символи алфавіту A розташовуються у комірках стрічки окремо,
– у порожній комірці завжди знаходиться порожній символ $o\in A$,
– над окремою коміркою може знаходитися вказівник (головка зчитування) з позначкою (∇).

Заповнені символами вхідного алфавіту комірки стрічки і положення вказівника над коміркою з символом a_i показані на рис. 4.10.

Конфігурацію стрічки визначає кортеж, побудований на послідовності розташування символів у комірках стрічки (зліва на право) з врахуванням положення вказівника. Так за даними рис. 4.10 конфігурація має вигляд:

$$(\ldots,a_k,\nabla a_i,\ldots,a_r,\ldots). \qquad (4.10)$$

Конфігурації (4.10) стрічки залежать від часу і визначають стани $z(t)\in Z$ автомату $\mathbb{A}$.

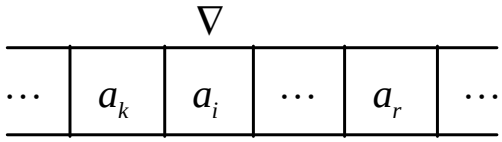


Рис. 4.10. Стрічковий автомат моделювання пам’яті

Домовимося, що початковому стану $z(t_0)$ автомату буде відповідати конфігурація стрічки, за якою вказівник знаходиться над першою не порожньою зліва коміркою. Якщо існує заключний стан автомату, тоді йому ставиться у відповідність конфігурація, в якій вказівник знаходиться над порожньою коміркою за останнім не порожнім символом вхідного алфавіту. Наприклад, конфігурація $(o,\ldots,o,\nabla a_2,a_1,o,a_2,a_3,\ldots)$ відповідає $z(t_0)$, а $(\ldots,a_4,a_6,o,o,a_1,\nabla o,\ldots,o)$ – визначає заключний стан автомату.

У кожний дискретний момент часу в залежності від положення вказівника (∇) комірки стрічки автомат виконує дії запису або читання, змінюючи при цьому стан автомату. Дія запису символу алфавіту A за функцією φ відбувається в комірці під вказівником (попередній символ затирається або зсувається, разом з послідуючими символами, на одну комірку та на його місце записується новий символ) і вказівник встановлюється над новою коміркою або залиша-

ється у тому ж положенні за визначенням функції станів. Отже функція станів задається відображенням $\varphi: A \times Z \to Z$. Так, якщо конфігурація стрічки в момент часу t_k $(\ldots, o, a_2, \nabla a_1, a_1, o, \ldots)$ і функція для цього моменту виконує запис за правилом:

$$\varphi(a_1, z(t_k)) : \nabla a_1 \mapsto a_3, \nabla *; \tag{4.11}$$

де * – вказує на переміщення вказівника на наступну комірку після комірки з символом a_3 (якщо * відсутня переміщення вказівника не відбувається), тоді отримаємо новий стан $z(t_{k+1})$, якому відповідає конфігурація $(\ldots, o, a_2, a_3, \nabla a_1, o, \ldots)$. Застосовуючи знову правило (4.11), приведемо автомат в заключний стан $z(t_{k+2})$ з конфігурацією $(\ldots, o, a_2, a_3, a_3, \nabla o, \ldots)$.

Дія читання за функцією $\psi: A \times Z \to A$ виконується в момент часу t_j так, що видаляється символ на стрічці під вказівником (зміст комірки витирається і послідовно символи, що розташовані справа або зліва від стертого символа зсуваються на одну комірку пам'яті, щоб не зосталося незаповнених комірок), показник встановлюється над новою коміркою або зостається на тому ж місці. Пояснимо застосування функції ψ на прикладі розглянутої конфігурації для моменту t_k $(\ldots, o, a_2, \nabla a_1, a_1, o, \ldots)$.

Введемо спочатку позначку $^{-}\nabla^{*}$ $(^{+}\nabla^{*})$ зсуву символів розташованих зліва (з права) від видаленого символа на одну комірку вправо (вліво) і встановлення показника над коміркою з символом *.

Нехай функція читання у цей момент діє за правилом:

$$\psi(a_1, z(t_k)) : \nabla a_1 \mapsto a_1, {}^{+}\nabla, \tag{4.12}$$

тоді станові $z(t_{k+1})$ буде відповідати конфігурація $(\ldots, o, a_2, \nabla a_1, o, \ldots)$ і на виході маємо символ $a_1 = \psi(a_1, z(t_k))$. Ще раз застосовуючи правило (4.12) до стану $z(t_{k+1})$ переведемо автомат у заключний стан з конфігурацією $(\ldots, o, a_2, \nabla o, \ldots)$. Отже, виходом автомату буде ланцюжок $a_1 a_1$.

За модель пам'яті ЕОМ частіше приймається напівобмежені стрічки, які зображають горизонтально або вертикально; причому вказівник не використовується, тому що обмежений край стрічки приймається за вершину (рис. 4.11). Відрахування комірок у таких

стрічках починається з вершини. Якщо дії запису і читання виконуються через вершину стрічки, тоді таку модель пам'яті звуть *магазином,* а відповідний автомат над пам'яттю – *магазинним автоматом* (МП автомат). Запис символа вхідного алфавіту відбувається через першу комірку (*вершину магазина*) за допомогою зсуву змісту комірок на одну комірку і у вивільнену комірку вершини заноситься відповідний символ. На рис. 4.11, *b* наведено результат запису символа a_k в магазин (рис. 4.11, *a*). При виконання дії читання символ у вершині магазину стирається і символи, котрі зосталися у магазині, зсуваються в сторону вершини на одну комірку. Результат читання символа a_i з магазину (рис. 4.11, *a*) наведено на рис. 4.11, *c.* Інколи дію запису в магазин коротко називають операцією «*в магазин*», відповідно дію читання – «*з магазину*». Зрозуміло, що зміст магазину в момент t_k визначає конфігурацію стрічки, яка відповідає стану $z(t_k)$ автомату.

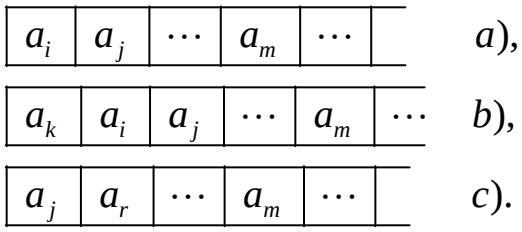


Рис. 4.11. Результати виконання дій «записати» і «читати» у МП автоматі

Різновидом магазину є *стек,* будь яка комірка якого доступна для читання зі збереженням його змісту. Стеки застосовують при організації формалізованих розборів мовних конструкцій, при організації процедур в програмуванні тощо. З іншими різновидами магазинної пам'яті можна ознайомитися в літературі з програмування, а з одним із біологічних автоматів над пам'яттю у цікавій монографії [14].

4.2.5. Запитання

1. Як визначається не детермінований автомат?
2. Що визначає напрямок поведінки не детермінованого автомату?
3. Як пов'язані шлях поведінки і напрямок поведінки не детермінованого автомату?
4. Які характеристики може мати шлях поведінки не детермінованого автомату?

5. Що таке повний шлях поведінки не детермінованого автомату?
6. Як визначається мова не детермінованого автомату?
7. Що означає «розшифровка мови» не детермінованого автомату?
8. В чому суть прийому розшифровки мови автомату?
9. Який автомат називається словарним?
10. Який автомат називається детермінованим?
11. Що моделюють автомати над пам’яттю?
12. Як технологічно представляються автомати над пам’яттю?
13. Як задати автомат над пам’яттю?
14. Які дії можуть виконувати автомати над пам’яттю?
15. Що таке конфігурація стрічки автомату над пам’яттю?
16. Чи можливо відтворити зміст стрічки за її конфігурацією?
17. Які одно стрічкові автомати Вам відомі?
18. Для яких автоматів над пам’яттю необхідно вводити вказівник комірки?
19. Що таке магазинна пам'ять?
20. Як організована стекова пам'ять?

4.2.6. Завдання і вправи.

1. Побудувати не детерміновані автомати, за якими утворюються мови:

а) $\{10^n1^m;\ n,m\in\mathbb{N}\}$,

б) $\{1^n2^k234;\ n,k\in\mathbb{N}\}$,

в) $\{ab1^k\,ab0^n;\ k,n\in\mathbb{N}\}$,

г) $\{(10)^k01(01)^n;\ k,n\in\mathbb{N}\}$,

д) $\{ab^kba^m;\ k,m=0,1,2,...\}$.

2. На станах автоматів знайдених у завданні 1 виділити суміжні множини.
3. Знайти напрямки поведінки автоматів завдання 1.
4. Знайти всі шляхи поведінки кожного автомату завдання 1.
5. Серед шляхів поведінки автоматів вправи 4 знайти повні шляхи.
6. Виконати декомпозицію шляхів поведінки автоматів, знайдених за вправою 4.

7. Скористатися результатами вправи 4 і побудувати графи автоматів за повними шляхами поведінки. Чи є побудовані графи графами автоматів?
8. Виконати розшифровку мов завдання 1.
9. Звести графи автоматів завдання 1 до простих шляхів.
10. Для наведених у завдання 1 мов побудувати детерміновані автомати.
11. Для наведеного на рис. 4.11 нескінченного автомату розв'язати завдання:

а) конвертувати ланцюжки:

$$l = 011001,\ l = 11100001,\ l = aaabaaa,\ l = 0b0b0b00;$$

б) стерти символ b у ланцюжках:

$$l = aabbaba,\ l = bb111b1,\ l = ab11ba01,\ b = 0 \text{ і } l = 100013;$$

в) транспонувати ланцюжки:

$$l = 01110000,\ l = 0256720,\ l = 0abc\text{о}1110,\ l = \text{о}abcdab\text{о};$$

г) перетворити ланцюжки:

$$l_1 = слон \text{ до } l_2 = сон,$$
$$l_1 = кухар \text{ до } l_2 = рахунок,$$
$$l_1 = соната \text{ до } l_2 = атаман$$

12. Записати послідовні конфігурації стрічки автомату при розв'язанні завдання 1.
13. Модифікувати функції φ і ψ автомату для простішого розв'язку завдання 1.
14. Розв'язати завдання 1 для автомату з магазинною пам'яттю.
15. Розв'язати завдання 1 для автомату із стековою пам'яттю.

16. Записати послідовні конфігурації стрічки при розв'язанні завдання 1.
17. Модифікувати функції φ і ψ автомату для простішого розв'язку завдання 1.
18. Розв'язати завдання 1 для автомату з магазинною пам'яттю.
19. Розв'язати завдання 1 для автомату із стековою пам'яттю.

4.3. Алгебри і структури

Апарат алгебр і структур застосовуються у теорії програмування для конструювання і аналізу формальних об'єктів: алгоритмів, граматик мов, графів, автоматів тощо.

4.3.1. Алгебри

Алгебра як математична дисципліна вивчає алгебраїчні операції. Наприклад, в рамках алгебр з'ясовуються їх групові властивості, універсальність та інше [1, 11, 12, 13, 15, 18].

Позначимо через B довільну множину з визначеною над нею множиною n операцій (сигнатуру) Σ, відповідна місність яких $m_1, m_2, \ldots, m_n$.

Визначення 4.3. Алгеброю Ξ типу $(m_1, m_2, \ldots, m_n)$ над множиною B звуть упорядковану пару, яка складається з цієї множини та сигнатури, тобто $\Xi = \langle B, \Sigma \rangle$.

В цілому алгебра Ξ не визначає „порядок-будову" множини B по операціям сигнатури Σ за виключення випадку, коли множина B упорядкована. Тобто, якщо між елементами множини B можливо ввести відношення порядку, наприклад, $(\leq)$ і сигнатуру визначити у вигляді

$$\Sigma = \{\min^2, \max^2\} \mid \{\bigcup^2, \bigcap^2\} \mid \{\vee^2, \wedge^2\},$$

тоді алгебра Ξ визначає упорядковану структуру – решітку $C = \langle B, \Sigma \rangle$ [15]. Тому алгебру Ξ інколи звуть алгебраїчною структурою.

Звернемо увагу на особливості алгебр за значенням результату операцій її сигнатури.

Позначимо через $(*)$ довільну операцію сигнатури $*^k \in \Sigma$ і визначимо її як відображення декартового добутку B^k у саму множину B, тобто

$$* : B^k \to B. \tag{4.13}$$

У випадку одномісного відображення ($k=1$) операцію, яка відповідає цьому відображенню звуть *підстановкою*. Результат дії підстановки зручно представляти у вигляді бінарного відношення між рядками елементів деякої множини. Наприклад, підстановка u_u на числовій множині $B=\{1,2,\ldots,r\}$ може бути визначена схемою

$$u=\begin{pmatrix}1 & 2 & 3 & 4 & 5\\ 3 & 1 & 2 & 5 & 4\end{pmatrix},$$

в якій зображено відношення цифр першого і другого рядків (1 переходе у 3, 2 переходе в 1 і так далі).

Визначення 4.4. Алгебра Ξ зветься *універсальною алгеброю*, якщо для будь якої операції її сигнатури виконується умова (4.13).

Наприклад, алгебра $\Xi=\langle\mathbb{N},\{+^2,\times^2\}\rangle$ є універсальною типу (2,2), а алгебра $\Xi=\langle\mathbb{N},\{+^2,-^2\}\rangle$ є *частково універсальною типу* (2,2) тому що для операції віднімання ($-$) частково виконується умова (4.13). Слід зауважити, що алгебра на вільній мові $\mathbb{F}(A)$ відносно операції конкатенації ($\boxtimes$) є універсальною алгеброю.

Визначення 4.5. Не порожня підмножина D множини B алгебри Ξ називається *замкненою відносно операції* $(*)$, якщо для цієї операції $*^k\in\Sigma$ виконується умова $*:D^k\to D$. Якщо ж підмножина D є замкненою відносно всіх операцій сигнатури Σ, тоді її звуть *замкненій в алгебрі* Ξ. Алгебру, побудовану на замкненій множині D, $\Xi_D=\langle D,\Sigma'\rangle$, $\Sigma'\subseteq\Sigma$ називають *підалгеброю* алгебри Ξ.

Зрозуміло, що підалгебра Ξ_D є універсальна алгебра і у випадку, коли $\Sigma'=\Sigma$, $D=B$ алгебра Ξ_D є *невласною підалгеброю* алгебри Ξ. Алгебра $\langle\{\varepsilon\},\{\boxtimes\}\rangle$, де ε – порожній ланцюжок є підалгерою універсальної алгебри $\langle\mathbb{F}(A),\{\boxtimes\}\rangle$ вільної мови $\mathbb{F}(A)$. Апарат підалгебр дозволяє виконати розбиття алгебри на класи еквівалентності та інше.

Якщо розглядати алгебри з позицій властивостей множини B відносно операцій сигнатури Σ, тоді можливо вести розмову відносно групових властивостей алгебр, гомоморфізмів алгебр.

Нехай $(*)$ деяка бінарна операція сигнатури Σ, яка визначена на множині B, тоді

Визначення 4.6. Групою відносно операції $(*)$ зветься алгебра Ξ, якщо для цієї операції виконуються умови:

1) операція $(*)$ всюди визначена, тобто для будь яких елементів $a,b\in B$ існує елемент $c\in B$ такий, що $a*b=c$;
2) операція $(*)$ асоціативна: для довільних елементів $a,b,c\in B$ має місце $(a*b)*c=a*(b*c)$;
3) у множині B існує одиничний елемент θ такий, що $a*\theta=\theta*a$, для будь якого елементу $a\in B$;
4) для будь якого елементу $a\in B$ існує у множині B обернений елемент a^{-1} такий, що $a^{-1}*a=a*a^{-1}=\theta$.

Група, в якій операція $(*)\equiv(+)$, зветься *адитивною групою* або *модулем*, а група, в якій – $(*)\equiv(\times)$, звуть *мультиплікативною*. Крім того, якщо відносно операції $(*)$ виконується властивість комутативності $a*b=b*a$, для будь яких елементів $a,b\in B$, тоді групу відносно цієї операції звуть *комутативною* або *абєлєвою*. Якщо для алгебри Ξ відносно операції $(*)$ виконуються перші три властивості, тоді таку алгебру відносно цієї операції називають *напівгрупою* (*адитивною напівгрупою*, *мультиплікативною напівгрупою*, *комутативною напівгрупою*).

Класичним прикладом адитивної і мультиплікативної абєлєвих груп є алгебра на множині дійсних чисел $\mathbb{R}$. Алгебра на вільній мові довільного алфавіту A відносно операції конкатенації є вільною напівгрупою, тому що четверта умова визначення 6 для операції $(\boxtimes)$ не має місця, роль одиниці θ виконує порожній ланцюжок $\varepsilon\in\mathbb{F}(A)$.

Подальший розвиток універсальних алгебр, на системі відношень (*множини моделей*) і аналогій між ними призвів до створення *алгебраїчних систем* [11].

Нехай завдана деяка множина B, на якій визначена: множина операцій Σ і множина відношень H, тоді

Визначення 4.7. Алгебраїчною системою Ω називається упорядкована трійка

$$\Omega=\langle B,\Sigma,\mathrm{H}\rangle.$$

Алгебраїчна система Ω є звичайною алгеброю, якщо множина H порожня і у разі, коли $\Sigma=\varnothing$ система Ω утворює *модель* (*реляційну систему*).

В межах алгебраїчних систем можливо відтворити формальну систему логіки висловлювань, наприклад, мови першого ступеню і інше [11].

Розвиток штучних систем теорії програмування призвів до створення нової теорії *систем алгоритмічних алгебр* [21]. Конструктивно система алгоритмічних алгебр будується на інформаційному просторі R абстрактної обчислювальної машини. Над цим інформаційним простором вводяться сигнатури абстрактних операторів $P, Q, \cdots$ та умов $\alpha, \beta, \cdots$ абстрактної машини, на яких задаються дві алгебри: *алгебра операторів* Ξ_o^0 і *алгебра умов* Ξ_u^0. Потім розглядається алфавіт $W^0 = \{\Xi_o^0, \Xi_u^0\}$ і задається сигнатура Σ_1 операцій: кон'юнкції $\alpha \wedge \beta$, диз'юнкції $\alpha \vee \beta$, заперечення $\neg\alpha$, множення операторів PQ, α-диз'юнкції операторів P і Q.

$$\underset{\alpha}{(P \vee Q)} \to \begin{cases} P(r), & \alpha(r) = 1; \\ Q(r), & \alpha(r) = 0; \\ \textit{невизначено}, & \alpha(r) = \mu, \quad r \in R; \end{cases}$$

α- ітерація оператора P

$$\underset{\alpha}{\{P\}} \to \begin{cases} P^i(r), \quad \alpha(P^i(r)) = 1, \quad i = 0,1,2,\ldots; \quad P^0(r) = r; \\ \qquad\qquad\qquad\qquad\qquad\qquad (P^j(r)) = 0, \quad j < i; \\ \textit{невизначено}, \quad \alpha(P^j(r)) = \mu; \quad r \in R; \end{cases}$$

і лівого множення умови α на оператор P – $P\alpha \to \alpha(P(r)), \quad r \in R$.

На парі W^0 і Σ_1 утворюються нові алгебри Ξ_o^1 і Ξ_u^1, потім над алфавітом $W^1 = \{\Xi_o^1, \Xi_u^1\}$ за допомогою сигнатури Σ_1 знову утворюються дві алгебри Ξ_o^2 і Ξ_u^2, і так далі. Тоді під *системою алгоритмічних алгебр* розуміється пара $\left\langle \bigcup_i \Xi_o^i, \bigcup_i \Xi_u^i \right\rangle$. Очевидно, що система алгоритмічних алгебр є породжувальною універсальною алгеброю. Найбільшого застосування системи алгоритмічних алгебр (САА) набули в питаннях теорії програмування, прикладній теорії алгоритмів та інше [1, 2, 19], САА покладені в основу науки «Алгоритміка» [18].

4.3.2. Формальні конструктивні структури

Як вказувалося раніше, розглянуті у розділі 1 формальні системи дозволяють будувати мовні конструкції предметних областей і не задають алгебраїчну структуру цих систем. З іншого боку, за межами визначення алгебраїчних структур, алгебраїчних систем та систем алгоритмічних алгебр знаходяться правила виконання операцій сигнатур і аксіоми застосування цих правил. Тому для «повноцінного» дослідження мов, мовних конструкцій, алгоритмів, програмних конструкцій та іншого необхідно застосовувати два різні математичні розділи: теорію формальних систем та алгебр. Слід зауважити, що класичну алгебраїчну структуру решітку у формальних системах використовувати не можливо. Наприклад, множину елементів вільної мови загалом не просто упорядкувати, хоча за теоремою Цермело будь яку множину можливо упорядкувати. Для формальних систем структуру слід розглядати як будову, до речі, структура (латинське слово structyra) також перекладається як будова [20]. Виходячи з цих міркувань введемо в розгляд конструктивний об'єкт *формальну структуру* [6 – 9, 22, 23], в межах якої, для заданої предметної області можливо створювати мовні, алгоритмічні, графічні та інші конструкції і проводити їх дослідження.

Визначення 4.8. Формальною конструктивною структурою називається упорядкована трійка – множини B (носія структури), сигнатури Σ і аксіоматики (числення структури) Λ, тобто

$$C_{Nc} = \langle B, \Sigma, \Lambda \rangle. \tag{4.14}$$

У сигнатуру Σ можуть включатися різні операції та відношення, а аксіоматика Λ може включати в себе загальні аксіоми, правила застосування операцій сигнатури, аксіоми застосування правил та інші форми. Слід зауважити, що за межами структури (4.14) може знаходиться вільна мова, побудована на множині $B \bigcup \Sigma$, до якої, при потребі, ми будемо звертатися.

Наведемо властивості формальної структури.

Властивість 4.1. Структура (4.14) задає формальну систему.

Очевидно, що структура (4.14) відтворює, як частковий випадок, за визначенням 4.1 формальну систему з відповідною вільною мовою $Nc^* \subset \mathbb{F}(B \bigcup \Sigma)$, операцією виводу $(\rightarrow)$ і аксіомами та правилами виводу з аксіоматики Λ структури (4.14).

Властивість 4.2. Формальна структура на множині B задає алгебру.

Властивість 4.3. Частковим випадком формальної структури є решітка.

Дійсно, нехай множина B упорядкована, а сигнатура утворена множиною операцій $\{\wedge^2, \vee^2\}$, тоді формальна структура з аксіомами відносно цих операцій:

$$a \wedge a = a, \qquad a \vee a = a,$$
$$a \wedge b = b \wedge a, \qquad a \vee b = b \vee a,$$
$$(a \wedge b) \wedge c = a \wedge (b \wedge c), \; (a \vee b) \vee c = a \vee (b \vee c),$$
$$a \wedge (a \vee b) = a, \qquad a \vee (a \wedge b) = a,$$

для будь яких елементів $a, b, c \in B$, утворює решітку [17].

Для прикладу застосування формальної структури розглянемо досить повчальну задачу побудови структури лінійного простору.

Приклад 4.2. Нехай задана деяка множина B і a, b її довільні елементи причому такі, що їх значення можуть дорівнювати нулеві – 0 або 1 – одиниці. Розглянемо декартовий добуток $B^n = E$, на якому визначені елементи $x, y, z, u, v \in E$. Побудуємо формальну структуру відтворення лінійного простору на множині E.

▷ Лінійний простір визначається на операціях складання (+) елементів множини E і множення (·) елементів множини B на елементи множини E, тому формальна структура має вигляд

$$C_E = \left\langle E, \{\rightarrow^2, +^2, \cdot^1\}, \Lambda \right\rangle.$$

Аксіоматика Λ буде складатися з аксіоми виводу

$$\sigma = x \,|\, y, \quad \sigma \in E;$$

правил виводу:

1) $\sigma \rightarrow \alpha \cdot \sigma$,

2) $\alpha \rightarrow a \,|\, b \,|\, 0 \,|\, 1 \,|\, (a + b)$;

і правил виконання операцій $(+), (\cdot)$:

$$x + y = u, \quad a \cdot x = v;$$

та аксіом застосування цих правил:

1) $x + y = y + x$;

2) $(x + y) + z = x + (y + z)$;

3) у множині E $_E$ існує елемент θ θ такий, що $0 \cdot x = \theta$;

4) $1 \cdot \sigma = \sigma$;

5) $(a + b) \cdot \sigma = a \cdot \sigma + b \cdot \sigma$;

6) $a \cdot (x + y) = a \cdot x + a \cdot y$;

7) $a \cdot b \cdot \sigma = a \cdot (b \cdot \sigma)$,

де у виразах 4), 5), 7) $\sigma = x \mid y$ є аксіома. $\triangleleft$

Як видно з розглянутого прикладу, операції складання та множення на множині B не визначені, але ця проблема розв'язується за аксіомами 5) і 7). Введена формальна структура C_E відтворює будову множини E, тобто за аксіомою і правилами виводів маємо, що E складається з лінійних конструкцій над простими конструкціями типу x, y за допомогою операцій складання та множення. Крім того структура C_E дозволяє вказати будову конструкцій-ланцюжків множини E. Так, наприклад, для ланцюжка

$$l = a \cdot (a \cdot x) + a \cdot (b \cdot y) + b \cdot (a \cdot x) + b \cdot (b \cdot y) \in E,$$

після застосування до аксіоми σ два рази першого правила виводу потім другого правила, п'ятої аксіоми та аксіоми виводу і так далі маємо структуру-будову виводу:

$$W(l) = (\sigma, \alpha \cdot \sigma, \alpha \cdot \alpha \cdot \sigma, \alpha \cdot (a + b) \cdot \sigma = \alpha \cdot (a \cdot \sigma + b \cdot \sigma) = \alpha \cdot (a \cdot x + b \cdot y),$$
$$(a + b) \cdot (a \cdot x + b \cdot y) = a \cdot (a \cdot x + b \cdot y) + b \cdot (a \cdot x + b \cdot y) =$$
$$= a \cdot a \cdot x + a \cdot b \cdot y + b \cdot a \cdot x + b \cdot b \cdot y = l)$$

Конструктивна структура породжує формальну мову L. Для розглянутої структури C_E такою мовою є

$$L = \beta \cdot (\beta \cdot \ldots \cdot (\beta \cdot x) \ldots) + \beta \cdot (\beta \cdot \ldots \cdot (\beta \cdot y) \ldots) + \ldots, \qquad (4.15)$$

де через β позначено довільні елементи множини $a \mid b \mid \ldots$, а – $x \mid y \mid \ldots$ довільні прості конструкції множини E.

Очевидно, формальна мова може бути породжена різними структурами. Наприклад, мова (4.15) також породжується конструктивною структурою з такими правилами виводу:

1) $\sigma \to \alpha \cdot \beta$,

2) $\beta \to \sigma$,

3) $\alpha \to a \,|\, b \,|\, 0 \,|\, 1 \,|\, (a+b)$.

Визначення 4.9. Структури $C_B^1 = \langle B, \Sigma, \Lambda_1 \rangle$ і $C_B^2 = \langle B, \Sigma, \Lambda_2 \rangle$ називаються *функціонально еквівалентними формальними структурами*, якщо вони породжують одну і ту ж мову.

Формальні структури C_B^1 і C_B^2 є частковий випадок *однотипових* структур.

Нехай на множині B задана формальна структура $C_B = \langle B, \Sigma, \Lambda \rangle$ і B_1 не порожня підмножина множини B. Структура $C_{B_1} = \langle B_1, \Sigma, \Lambda \rangle$ однотипова до структури C_B, якщо має місце умова $\Sigma_1 \subseteq \Sigma$.

Визначення 4.10. Множина B_1 називається *замкненою*, якщо для будь якого елемента $b \in B_1$ і будь якої операції $(*)$ сигнатури Σ має місце $*(b) \in \mathbb{F}(B_1 \bigcup \Sigma)$.

Очевидно, що розглянутий вище лінійний простір є замкненим. Розглядаючи тепер на замкненій множині B_1 лінійного простору формальну структуру $C_{B_1} = \langle B_1, \Sigma, \Lambda \rangle$, де $\Sigma_1 \subseteq \Sigma$ і Λ_1 – частина аксіоматики Λ $(\Lambda_1 \subseteq \Lambda)$ відносно сигнатури Σ_1, отримаємо замкнену однотипову структуру до формальної структури C_B.

Визначення 4.11. Однотипову формальну структуру C_{B_1} до структури C_B назвемо *формальною підструктурою структури* C_B.

Наведемо ще одне досить важливе визначення для однотипових формальних структур C_{B_1} і C_B.

Визначення 4.12. Однотипові формальні структури C_{B_1} і C_B *ізоморфні*, якщо для будь яких елементів $b \in B$ і будь яких операцій $(*) \in \Sigma$, $(*_1) \in \Sigma_1$ та форм (продукцій, аксіом, властивостей) аксіоматик $\varphi \in \Lambda$, $\varphi_1 \in \Lambda_1$ існує взаємно однозначне відображення ρ множини B на множину B_1 таке, що

$$\rho(*(b)) = *_1(\rho(b)), \quad \rho(\varphi(b, *(b))) = \varphi_1(\rho(b), *_1(\rho(b))).$$

Слід звернути увагу на те, що введене поняття формальної структури можливо узагальнити додавши до сигнатури операцій Σ множину відношень H і розширити аксіоматику Λ. Таким чином будемо мати *систему формальної структури*

$$\mathbb{S}_B = \langle B, \Sigma, \mathrm{H}, \Lambda \rangle.$$

Так за допомогою логічних зв'язок $Z = (\wedge, \vee, \neg)$ можливо задати структури, які будують формули числення логіки висловлювань (дивися п. 1.4.2) або будують мову *замкнених формул* (не містять вільних змінних) логіки висловлювань (дивися п. 1.4.1). Якщо алфавіт B складається з вільних логічних змінних $\alpha_1, \alpha_2, \ldots, \alpha_n$ допоміжних символів $\sigma, (,)$ і множини зв'язок Z, сигнатура з однієї операції заміщення $(\rightarrow^2)$, а аксіоматика визначається за допомогою правил:

$$p_0 = \{\sigma \rightarrow \neg\sigma,\ \sigma \rightarrow (\sigma \wedge \sigma),\ \sigma \rightarrow (\sigma \vee \sigma),\ \sigma \rightarrow \alpha_i;\ i = 1, 2, \ldots, n\},$$

тоді формальна структура C_B відтворює мову формул числення висловлювань.

Нехай алфавіт B складається із предметних змінних α_i, H множина відношень зв'язок і аксіоматика формальної конструктивної структури

$$\Lambda = \begin{cases} p_0, \\ \textit{аксіоми при } \alpha = \alpha_i, \beta = \alpha_j, \gamma = \alpha_k;\ i, j, k \in \{1, 2, ..., n\}: \\ \alpha \wedge \alpha \sim \alpha,\ \alpha \vee \alpha \sim \alpha,\ \neg\neg\alpha \sim \alpha, \\ \alpha \wedge \beta \sim \beta \wedge \alpha,\ \alpha \vee \beta \sim \beta \vee \alpha,\ \neg(\alpha \wedge \beta) \sim \neg\alpha \vee \neg\beta, \\ (\alpha \wedge \beta) \wedge \gamma \sim \alpha \wedge (\beta \wedge \gamma),\ (\alpha \vee \beta) \vee \gamma \sim \alpha \vee (\beta \vee \gamma), \\ \neg(\alpha \vee \beta) \sim \neg\alpha \wedge \neg\beta,\ \alpha \wedge (\alpha \vee \beta) \sim \alpha,\ \alpha \vee (\alpha \wedge \beta) \sim \alpha, \\ \alpha \wedge (\beta \vee \gamma) \sim (\alpha \wedge \beta) \vee (\alpha \wedge \gamma),\ \alpha \vee (\beta \wedge \gamma) \sim (\alpha \vee \beta) \wedge (\alpha \vee \gamma). \end{cases}$$

тоді отримаємо систему формальної структури $\mathbb{S}_B$, котра породжує мову замкнених формул логіки висловлювань, за якими згідно табл. 1.1 можливо підрахувати їх значення.

4.3.3. Завдання і вправи

1. Довести, що наведені відношення завдають відношення порядку:
   а) на множині посад $\{a,b,\ldots\}$, посада b є підлеглою відносно a;
   б) на множині виробів $\{a,b,\ldots\}$, a важче за b:
   в) на множині номерів будинків $\{a,b,\ldots\}$, a не перевищує номеру b;
   г) на множині інтервалів $\{a,b,\ldots\}\subset I(\mathbb{R})$, a розташований далі ніж інтервал b.
2. Знайти добуток uv і vu підстановок:

$$u=\begin{pmatrix}1&2&3&4&5&6&7\\3&7&1&2&4&5&6\end{pmatrix} \text{ та } v=\begin{pmatrix}1&2&3&4&5&6&7\\2&1&2&3&1&7&3\end{pmatrix}.$$

3. З'ясувати, чи утворюють групу множини відносно операцій:
   а) раціональні числа з операцією множення;
   б) додатні раціональні числа з операціями ділення та множення;
   в) квадратні матриці з цілими елементами, детермінант яких дорівнює ±1, відносно операції множення матриць;
   г) множина обертів тримірного простору навколо його фіксованої точки з операцією множення, якщо за множення обертів прийняти їх послідовність виконання.
4. Довести, що якщо у напівтрупі існує правий одиничний елемент і для кожного її елемента існує правий обернений, тоді напівтрупа утворює групу.
5. Чи утворюється абєлєва група на вільній мові над алфавітом $A=\{a,\text{о}\}$ відносно операції конкатенації?
6. Довести, що вільна мова на будь якому алфавітові є замкненою відносно операції конкатенації.
7. Задати формальну структуру на вільній мові над алфавітом:
   а) $A=\{a,b,c\}$;
   б) $A=\{a,\text{о}\}$;
   в) $A=\{a,+,\text{о}\}$.

8. Створити систему формальної структури для числення висловлювань над алфавітами $A=\{\alpha,\beta,\gamma\}$, $B=\{(,),\to\}$ і $Z=\{\wedge,\vee,\neg,\Rightarrow\}$.

Вказівка: скористатися аксіомами підрозділу 1.6.1.

4.4. Підсумковий коментар

Висвітлені у цьому розділі поняття абстрактних моделей автоматів і структур мають безпосереднє відношення до теоретичних і практичних питань програмування. Мови програмування відносяться до класу контекстно-вільних конструктивних граматичних структур і тому за тезою Поста можуть бути представлені автоматом Т'юрінга, який функціонує над нескінченною стрічкою пам'яті за правилами $a_i z_j \to a_k z_p h$; $a_i, a_k \in A$, $z_j, z_p \in Z$, h – команда управління рухом стрічки (зсув на одну комірку вліво або вправо, або зсуву немає). Далі для алгоритму і формою його представлення програмою завжди можна побудувати автомат Т'юрінга (теза Т'юрінга), що дозволяє застосовувати дослідження теорії автоматів до аналізу і проектування програм. Зокрема, дати відповідь на важливе питання існування алгоритму і відповідної програми та інше.

Розглянутий апарат алгебр і їх модифікація – системи алгоритмічних алгебр [21] є важливий елемент моделювання та аналізу, як мікропрограм так і алгоритмічних програм. А розвинена на алгебрах Глушкова наука «Алгоритміка» [18] пропонує методики досліджень алгоритмів і алгоритмічних програм.

Запропоновані і розглянуті у розділі формальні конструктивні структури є новим математичним конструктивним апаратом, який поєднує елементи класичних алгебр, математичних структур, і конструктивної математики [9, 22, 23]. Цей апарат знайшов застосування в формальних граматиках, теорії алгоритмів, теорії графів та інших як інструмент для досліджень, конструювання, вирішення задач відтворення тощо. З останніми науковими дослідженнями по застосуванню конструктивних структур можна ознайомитися в наукових статтях співробітників кафедри КІТ Дніпропетровського національного університету залізничного транспорту.

Література до розділу 4

1. *Андон Ф.И., Дорошенко А. Е., Цейтлин Г. Е., Яценко Е. А.* Алгебро-алгоритмические модели и методы параллельного программирования. – К.: Академпериодика, 2007. – 634 с.
2. *Глушков В. М.* Синтез цифровых автоматов. М.: Физматгиз. 1962. – 476 с.
3. *Биркгоф Г.* Теория структур. М.: ИЛ, 1952. – 408 с.
4. *Брауэр В.* Введение в теорию конечных автоматов. – М.: Радио и связь, 1987. – 392 с.
5. *Бурбаки Н.* Теория множеств. – М.: Физматгиз, 1965. – 455 с.
6. *Ільман В. М.* Властивості формальних структур та їх підструктур // Вісник Дніпропетровського національного університету імені академіка В. Лазаряна, – 2007. – Вип. 14. – С. 99–104.
7. *Ильман В. М., Разумов С. Ю.* Структурный подход в формальных системах // Проблеми математичного моделювання. Міжнар. науково-метод. конфер. Тези доповідей. Дніпродзержинськ:ДДТУ, – 2005. – С. 147-148.
8. *Ільман В. М., Скалозуб В. В., Шинкаренко В. И.* Некоторые приложения формальных структур // Современные информационные технологии на транспорте, в промышленности и образовании. Тезисы международной научно-практической конференции. Днепропетровск.: ДНУЖТ, – 2007. – С. 63.
9. *Ільман В.М. , Скалозуб В.В., Шинкаренко В.І.* Формальні структури та їх застосування. Монографія. – Д.: ДНУЗТ, 2009. – 205 с.
10. *Катленд Н.* Вычислимость. Введение в теорию вычислимых функций. – М.: Мир, – 1983. – 256 с.4.
11. *Мальцев А. И.* Алгебраические системы. – М.: Наука, 1970. – 391с.
12. *Мозговой М. В.* Классика программирования: алгоритмы, языки, автоматы, компиляторы. Практический подход. – СПб.: Наука и Техника, – 2006. – 320 с.
13. *Ноден П., Китте К.* Алгебраическая алгоритмика. – М.: Мир, 1999. – 720 с.
14. *Паун Г., Розенберг Г., Саломаа А.* ДНК-компьютер. Новая парадигма вичислений: Пер. с англ. – М.: Мир, – 2003. 528 с.
15. *Скорняков Л.А.* Элементы теории структур. 2-е изд., доп. М.: Наука, 1982 – 160 с.
16. *Трахтенброт Б. А.* Алгоритмы и вычислительные автоматы. – М.: Сов. Радио, – 1974. – 200 с.

17. *Трахтенброт Б. А., Барздинь Я. М.* Конечные автоматы (поведение и синтез). – М.: Наука, – 1976. – 400 с.
18. *Цейтлин Г. Е.* Введение в алгоритмику. Киев: Сфера, 1998. – 310с.
19. *Ющенко К. Л., Суржко С. В., Цейтлін Г. О., Шевченко А. І.* Алгоритмічні алгебри. Навчал. посібник. – К.:ІЗЗМ, 1997. – 480 с.
20. *Яглом И. М.* Математические структуры и математические моделирование. – М.: Сов. радио, 1980. – 144 с.
21. *Глушков В. М., Цейтлин Г. Е., Ющенко Е. Л.* Алгебра. Языки. Программирование. – К.: Наукова думка, 1989. – 376 с.
22. *Шинкаренко В. И., Ильман В. М.* Конструктивно-продукционные структуры и их грамматические интерпретации. I Обобщенная формальная конструктивно-продукционная структура. // Кибернетика и системный анализ. – 2014. – Том 50, № 5. – С. 8-16.
23. *Шинкаренко В. И., Ильман В. М.* Конструктивно-продукционные структуры и их грамматические интерпретации. II Уточняющие преобразования. // Кибернетика и системный анализ. – 2014. – Том 50, № 6. – С. 15-28.

Розділ 5.
Мови програмування і формальні граматичні структури

Розглянуті у першому розділі посібника елементи формальної теорії тепер застосовуються до абстрактних і штучних граматик, при цьому використовується введене поняття конструктивної структури, що дає можливість коректно з формального боку будувати штучні мови і вирішувати змістовні задачі програмної інженерії. Оволодіння знаннями цього розділу дозволить глибше розібратися з базовим поняттям інженерії «мова програмування» і її застосуванням. В предметній області розділу використовуються поняття: *формалізму*, *формальної граматики*, *формальної мови*, *формальної структури*, та інші.

5.1. Мета розділу

Формальні граматики і їх структури відносяться до у універсальних методів моделювання предметних областей. Зокрема, за допомогою формальних граматик моделюються, розробляються та конструюються алгоритмічні мови програмування, транслятори, алгоритми, штучні автомати тощо.

Виходячи з цього метою розділу є надання необхідні знань з формальних граматик та граматичних структур, їх типів, прийомів конструювання мовних ланцюжків (програм) і навичок граматичного моделювання та застосування результатів моделювання. Ознайомити читача з сучасним положенням граматичних структур та формальних граматико-подібних систем, надати відомості з ідеології структури мов програмування, що покращить рівень підготовки студентів за напрямом програмної інженерії.

5.2. Виникнення і розвиток формальних мов

Кожен з нас уявляє собі те, які виникають проблеми перед людиною, коли вона не володіє *природною* мовою на якій спілкуються оточуючі її люди. Також кожний, хто хоча б один раз сидів за терміналом обчислювальної машини (ЕОМ), знає, що машина не бажає спілкуватися з користувачем, якщо він порушує правила тієї *штучної* мови, яку вона «розуміє». Зрозуміло, що висловлювання «володіє мовою», «розуміє мову» є побутовими не чіткими тому, що, як відомо з розділу 1, мова може складатися з нескінченної кількості слів, якими в принципі не можливо оволодіти. Коректніше було б говорити про володіння (розуміння) граматикою природної (штучної) мови, бо основою граматик є скінчені множини елементів алфавітів і правил, якими людина в змозі оволодіти. Отже, для коректного розв'язання проблем спілкування необхідно володіти граматиками відповідних мов. Але виникає питання: «чи не можливо обійтися природними мовами для «спілкування» з ЕОМ?». Відповідь поки що є негативною тому, що природні мови *синтаксично* не однозначні. Наприклад, в залежності від того на якому предметі столі або олівцеві концентрується увага, можна чути фрази: «на столі лежить олівець» або «олівець лежить на столі». Змістовно (*семантично*) ці висловлювання однакові, але конструктивно це різні словосполучення. Крім того, системи морфологічних і синтаксичних правил природних мов інколи не чітко визначені.

Синтаксична неоднозначності, зайвості, не логічність граматик природних мов підштовхнули польського лікаря Людовіка Зоменгофа у 1887 р. створити мову есперанто. В граматиці цієї мови всього шістнадцять загально граматичних правил та два відмінки (називний і знахідний), досить прості правила словосполучень. Але недоречності природних мов не зупиняли дослідників двадцятого століття в напрямку формалізації мов [11] і призвели до виникнення математичної лінгвістики [4]. Математична лінгвістика знайшла своє застосування в автоматизації процесів переробки інформації (розробці автоматизованих перекладачів) та стала основою при розробці теорії штучних мов. Штучні мови виникли в першу чергу, як інструмент для опису різних предметних областей людських знань геометрична мова, фізична мова, алгоритмічна мова і інші. В подальшому потреба в них виникла при створені ЕОМ, як мов для діалогу людини з машиною. Мабуть першою такою штучною мовою для спілкування з ЕОМ

«МИР» була мова «Аналітик» [5], розроблена Інститутом кібернетики Академії Наук України у 1968 р.

Серед штучних мов слід виділити клас формальних мов, для опису яких існують точні правила виводів конструкцій – ланцюжків. Формальні мови є основою для створення *алгоритмічних мов*, призначених для запису *алгоритмів*, а також *мов програмування*, які застосовуються для описування інформації та програм-алгоритмів обробки на ЕОМ. Зрозуміло, що мови програмування стають «досяжними» для програмістів у реалізації на ЕОМ після створення відповідних *трансляторів* з цих мов. Таким чином зв'язок між цими мовами можливо представити у вигляді схеми (рис. 5.1).

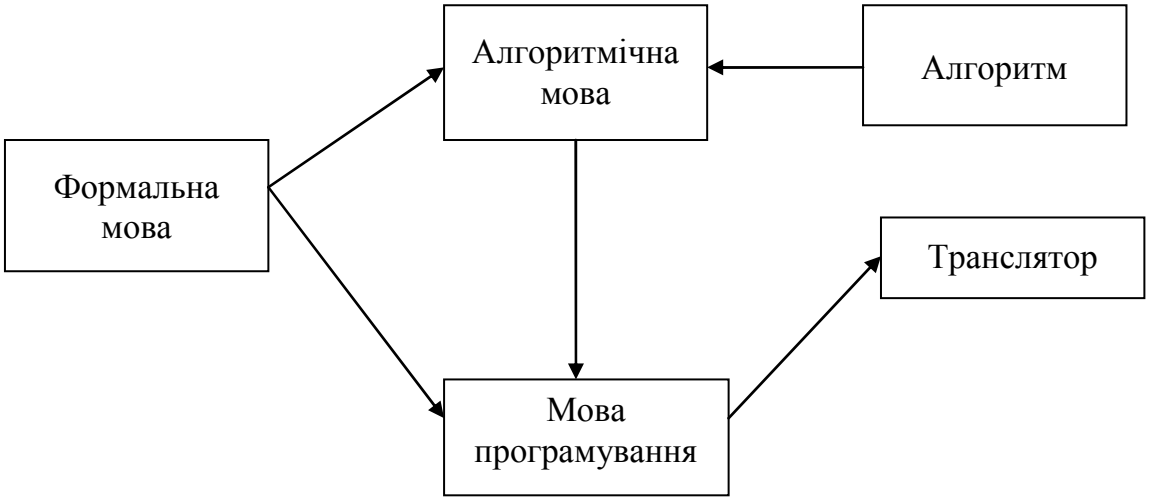


Рис. 5.1. Зв'язок між мовами

Теорія формальних мов виникла на стикові математичної логіки, алгебри та теорії алгоритмів і остаточно сформувалась як наука в п’ятидесятих роках двадцятого сторіччя. Вагомий внесок у її розвиток внесли вчені: А. Туе, Н. Хомський [13], А. В. Гладкий [4], С. Гінзбург [3] і інші. Огляд різних підходів формалізації предметних областей з позицій граматик можна знайти в роботі [12].

Серед формальних мов виділимо класи: *породжених мов*, тобто мов, в граматиках яких дозволяється вивести (породити) *правильні* ланцюжки мов і не можливо вивести ні одного не правильного ланцюжка; *розпізнавальних мов*, граматики яких дозволяють розпізнати правильні ланцюжки формальної мови. Клас породжених мов є більш розвиненим і жаданим на практиці тому, що за допомогою синтаксичного аналізу, в межах цього класу, можливо з’ясувати правильність ланцюжка, крім того сучасні мови програмування відтворені на класі породжених формальних мов. Таким чином на класі породжуючих граматик досліджу-

ються можливості побудови класів формальних породжених мов, розпізнання структур ланцюжків цих мов, алгоритмічної розв'язності, властивості мов та інші.

5.3. Формальні структури породжуючих граматик

5.3.1. Загальні поняття

Розглянемо деяку предметну область R, неподільні об'єкти (дослідник області R самостійно їх визначає), якої позначимо різними малими символами латинського алфавіту, при потребі з нижніми індексами, і на цій скінченій послідовності символів утворимо алфавіт A. Алфавіт A назвемо *термінальним алфавітом*, тобто термінальний алфавіт задає об'єкти предметної області R. Алфавіт може бути однорідним або неоднорідним, в останньому випадку його символи мають атрибут, представлений змістовно чи іншим чином.

У кожній предметній області використовується та чи інша метатермінологія (назви об'єктів, груп об'єктів, їх характеристики і інше), на яких введемо *метамовний словник не терміналів* N. Елементи словника, абстрагуючись від змісту не терміналів, будемо позначати символами грецького алфавіту з індексами або без них. Таким чином маємо *не термінальний алфавіт* N метамовних значень. Для прикладу за предметну область R візьмемо учнівський зошит, який складається, для спрощення, з обгортки та аркушів; тоді термінальний алфавіт $A=\{a,b\}$, де позначено: a – обгортка, b – аркуш; не термінальний алфавіт $N=\{\sigma,\alpha,\beta\}$, де метамовні позначення σ – «учнівський зошит», α – «обгортка», β – «аркуш». Об'єднання алфавітів A і N назвемо *алфавітним словником* $V=A\cup N$ або просто *словником*. Тепер можливо визначити формальну граматику.

Визначення 5.1. Формальною породжуючою граматикою G, визначеною на термінальному алфавітові A і не термінальному алфавітові N з *початком (початковою множиною)* $U\subseteq N$, $o\notin U$ (o – порожній символ) та *системою правил виводу* P називається упорядкована четвірка

$$G=\langle A,N,U,P\rangle. \tag{5.1}$$

В теорії формальних граматик система виводу P , завжди складається із вільного (неупорядкованого) скінченого кортежу *продукцій* або правил. Кожна *продукція* $p_i \in P$ має вигляд заміщення $p_i : x_i \to y_i$ (p_i – є бінарним відображенням, за яким елемент x_i заміщується – y_i), в якому ліва і права частини $x_i, y_i \in \mathbb{F}(V)$, а символ $\to$ є зовнішнім по відношенню до алфавітів граматики G. Під початком $U \subseteq N$ в граматиці розуміється те, що будь яка продукція $p_i \in P$, ліва частина якої складається з довільного не термінала підмножини U, повинна застосовуватися першою при виводі ланцюжка; такі продукції назвемо *початковими продукціями*. Пояснимо це на прикладі учнівського зошита, для якого задамо грамматику

$$G = \langle \{a,b\}, \{\sigma, \alpha, \beta\}, \sigma, \{p_1 : \sigma \to a\alpha,\ \ p_2 : \alpha \to b\alpha,\ \ p_3 : \alpha \to b\} \rangle \qquad (5.2)$$

У граматиці (5.2) з початком σ вивід будь якого ланцюжка формальної мови повинен починатися з застосування початкової продукції p_1.

Слід звернути увагу на те, що порядок застосування продукцій системи P у формальній граматиці (5.2) довільний, але кожна послідовність продукцій виводу ланцюжка повинна починатися з застосування будь якої початкової продукції.

Зрозуміло, що граматика (5.1) визначена загальним чином, а граматика (5.2) задана конкретно через визначення елементів четвірки (5.1) і правил застосування продукцій. Граматика (5.1) задає систему числення по визначеній операції заміщення на словникові V і тому є можливість задати формальну структуру.

Визначення 5.2. Формальною граматичною конструктивною структурою $C(G)$ граматики G назвемо упорядковану трійку

$$C(G) = \langle V, \Sigma, \Lambda \rangle, \qquad (5.3)$$

де V – алфавітний словник (носій структури), Σ – сигнатура містить символи операцій, аксіоматика (числення) Λ складається з системи правил граматики P, в якій виділяються початкові продукції пов'язані з початком U граматики, тобто $\sigma \to x_j,\ \sigma \in U,\ x_j \in \mathbb{F}(V)$ – *аксіоми початкового виводу* або *аксіоми початку* та *аксіоми виводу*, продукції яких безпосередньо заміщують ланцюжки з термінальних і не термінальних символів на термінальні ланцюжки: $x \to a,\ x \in \mathbb{F}(V),\ a \in \mathbb{F}(A)$.

Граматичну структуру $C(G)$ і її граматику G назвемо *повною*, якщо усі символи словника застосовуються в аксіоматиці і числення не містить інших символів. Так для повної граматики (5.2) аксіоматика має вигляд

$$\Lambda = \begin{cases} p_3 : \alpha \to b \mid \alpha = b - \textit{аксіома виводу}, \\ p_1 : \sigma \to a\alpha - \textit{аксіома початку}, \\ p_2 : \alpha \to b\alpha; \\ \textit{аксіома} : \textit{послідовність застосування продукцій довільна}; \end{cases} \tag{5.4}$$

де застосовано позначення (| - «або») з граматичних правил у формі Бекуса – Наура і сигнатура $\Sigma = \{\to^2\}$

Введення формальної граматики у вигляді (5.3) коректніше ніж у вигляді (5.1), хоча б з того, що у структурі «все» задано і нема «зовнішніх» символів. Тому в подальшому при необхідності будемо користуватися для породжуючих граматик їх структурою (5.3). В межах формальної граматичної структури (5.3), визначаючи словник, сигнатуру і аксіоматику можна задавати різні породжуючі граматики G_i і їх структури $C(G_i)$.

Визначення 5.3. Структуру $C(G_i)$ конкретно заданої граматики G_i назвемо *структурою граматики* G_i.

Отже, граматична структура утворює клас структур граматик $C(G)\,C(G)$ такий, що $C(G_i) \subseteq C(G)\,C(G_i) \subseteq C(G)$, повну структуру якої завжди можливо однозначно відтворити за її аксіоматикою, чим в подальшому також будемо користуватися.

Приклад 5.1. Побудуємо конструктивну структуру граматики представлення алгоритмів у вигляді схем Насі – Шнайдермана.

▷ Відомо, що схеми Насі – Шнайдермана [1] складаються з чотирьох блоків: слідування, розгалуження і двох блоків циклу з «перед умовою» та «після умовою». Введемо термінальні позначки цих блоків: a – слідування, b – розгалуження, c – цикл з «перед умовою», d – цикл з «після умовою». Нехай нетермінал σ відповідає метафразі «схема Насі – Шнайдермана», тоді маємо словник $V = \{a, b, c, d, \sigma\}$. Сигнатуру визначимо на п’ятьох операціях $\Sigma = \{\to^2, \boxtimes^2, \circ^2, \circ_1^2, \circ_0^2\}$ і аксіоматику конструктивної структури задамо як

$$\Lambda = \begin{cases} \sigma \to a \boxtimes \sigma,\ \sigma \to a; \\ \sigma \to b \circ_1 \sigma,\ \sigma \to b \circ_0 \sigma,\ \sigma \to b \circ_1 \sigma \circ_0 \sigma; \\ \sigma \to c \circ \sigma,\ \sigma \to d \circ \sigma; \\ \sigma \to b \boxtimes \sigma,\ \sigma \to c \boxtimes \sigma,\ \sigma \to d \boxtimes \sigma; \\ \textit{операція } \to, \textit{не комутативна і не асоціативна}; \\ a \boxtimes b = ab,\ a \boxtimes b \boxtimes c = (a \boxtimes b) \boxtimes c = a \boxtimes (b \boxtimes c) = abc; \\ c \circ b \circ d = c \circ (b \circ d); \\ \textit{альтернативи вкладень } c \circ a \,|\, b \,|\, c \,|\, d\ ,\ b \circ_1 a \,|\, b \,|\, c \,|\, d\ ; \end{cases}$$

5.3.2. Виводи у класичних граматиках

Перейдемо до розгляду класичних формальних граматик. Зауважимо, що у класичній теорії формальних граматик сигнатура завжди має вигляд $\Sigma = \{\to^2\}$.

Досі використовувалося «інтуїтивне» поняттям виводу ланцюжка, дамо тепер формальне визначення виводу. Нехай задані будь які два ланцюжки x і y вільної мови $\mathbb{F}(V)$ побудованої на словникові V, тобто $x, y \in \mathbb{F}(V)$.

Визначення 5.4. Ланцюжок y зветься *безпосередньо виведеним* у структурі граматики $C(G)$ з ланцюжка x, $x \mapsto y$, якщо існує така продукція $p_i \in \Lambda$, після застосування якої до ланцюжка x отримаємо ланцюжок y, тобто $x \xrightarrow{p_i} y$. Або конкретно безпосередній вивід $x \mapsto y$ має місце тоді і тільки тоді, якщо $x = u_1 v_1 u_2$, $y = u_1 v_2 u_2$; де $u_1, u_2, v_1, v_2 \in \mathbb{F}(V)$ і існує продукція або аксіома $p_i \in \Lambda$ така, що $p_i : v_1 \to v_2$.

Таким чином безпосередній вивід $x \mapsto y$ супроводжується заміщенням підланцюжка ланцюжка x правою частиною продукції або аксіоми p_i. Наприклад, для формальної граматичної структури з аксіоматикою (5.4) маємо безпосередні виводи: $a\alpha \mapsto ab$, $a\alpha \mapsto ab\alpha$ і так далі.

Безпосередній вивід ланцюжка y з ланцюжка x $(x \mapsto y)$ визначає двохмісну операцію, яка є взагалі не комутативною і не асоціативною, вільна мова $\mathbb{F}(V)$ відносно неї є частково замкненою тому, що не до всякого ланцюжка $x \in \mathbb{F}(V)$ можливо застосувати вивід в струк-

турі Множина ланцюжків $\mathbb{F}(V)$ на цій операції утворює часткову універсальну алгебру. Виходячи з цього можливо стверджувати, що клас безпосередньо виведених ланцюжків $BWC(G)=\{y: x \mapsto y,\ x, y \in \mathbb{F}(V)\}$ в структурі $C(G)$ є підмножиною вільної мови $\mathbb{F}(V)$.

Визначення 5.5. Ланцюжок y зветься *виведеним* з ланцюжка x у формальній структурі $C(G)$, $x \Rightarrow y$ тоді і тільки тоді, якщо: $x = y$ або існує така послідовність ланцюжків $(z_0, z_1, \ldots, z_s) \subset \mathbb{F}(V)$, що $z_0 = x,\quad z_s = y$ і $z_{i-1} \mapsto z_i,\ i = 1,2,\ldots,s$.

Послідовність ланцюжків, які відповідають виведенню $x \Rightarrow y$ називають *виводом ланцюжка* y, $W(y) = (x, z_1, z_2, \ldots, z_s = y)$ в формальній структурі $C(G)$., а число s – *довжиною виводу* і позначають це так $|W(y)| = s$. Множина виведених ланцюжків $\{y: x \Rightarrow y;\ x, y \in \mathbb{F}(V)\}$ у формальній структурі утворює клас виведених ланцюжків $WC(G) \subseteq \mathbb{F}(V)$ в $C(G)$.

З визначення 5.4 маємо, що до класу виведених ланцюжків $WC(G)$ в структурі $C(G)$ відносяться також ланцюжки лівих і правих частин продукцій та аксіом аксіоматики цієї структури і безпосередньо виведені з них послідовності ланцюжків. Так для структури граматики з аксіоматикою (5.4) виведеним є ланцюжок α, $\alpha \Rightarrow \alpha$, за аксіомою виводу $\alpha = b$ або продукцією $\alpha \to b\alpha$, а також $\sigma \Rightarrow abb\alpha$, отже існує послідовність безпосередньо виведених ланцюжків $(\sigma, a\alpha, ab\alpha, abb\alpha = ab^2\alpha)$. Очевидно, виведення ланцюжка y з x: $x \Rightarrow y$ є бінарним відношенням, яке задовольняє властивостям рефлективності та транзитивності; отже відносно операції $(\Rightarrow^2)$ клас $WC(G)$ рефлексивно – транзитивно замкнений і між класами $BWC(G)$ і $WC(G)$ існує

Твердження 5.1. Клас $BWC(G)$ є підкласом класу $WC(G)$.

Розглянемо можливі виведення у класі $WC(G)$.

Виведення $x \Rightarrow y$ є *тупиковим* якщо йому відповідає вивід $W(y) = (x, z_1, z_2, \ldots z_s)$ такий, що до заключного ланцюжка $z_s \in WC(G)$ не можливо застосувати аксіоми або продукції аксіоматики структури $C(G)$. Наприклад, для наведеної структури з аксіоматикою (5.4) виведення $a\alpha \Rightarrow ab^3$ – тупикове тому, що вивід $(a\alpha, ab\alpha, abb\alpha, ab^3)$ не можливо продовжити і не існує аксіом або продукцій, які можливо застосувати для безпосереднього виводу з ланцюжка $abbb$.

Виведення $x \Rightarrow y$ є *повним* у структурі $C(G)$, якщо ланцюжок y складається тільки з термінальних символів алфавіту A, тобто $y \in \mathbb{F}(A)$. Таким чином повний вивід завершується обов'язково застосуванням аксіом виводу з Λ структури $C(G)$. Нескладно бачити, що виведення $a\alpha \Rightarrow ab^3$ є повне і тупикове в аксіоматиці (5.4).

Виведення $\alpha \Rightarrow y$ є *правильним*, якщо воно задовольняє умовам:

а) α – початковий символ, тобто $\alpha \in U \subseteq N$;

б) виведення $\alpha \Rightarrow y$ – тупикове;

в) виведення $\alpha \Rightarrow y$ – повне.

Ланцюжок y, який відповідає правильному виведенню $\alpha \Rightarrow y$ звуть *правильним ланцюжком* породженим у формальній структурі $C(G)$. Нескладно побачити, що ланцюжки: $ab, abb, abb, \ldots$ – правильні за аксіоматикою (5.4). Тепер можливо дати визначення формальної мови породженої у формальній структурі граматики G.

Визначення 5.6. Формальною мовою $L(G)$ породжену у формальній структурі $C(G)$ називають множину правильних ланцюжків.

Введене за визначенням 5.2 поняття формальної структури не передбачає обмежень на праві частини продукцій, тобто серед аксіом виводу аксіоматики можуть бути продукції типу $x \to o,\ x \in \mathbb{F}(V),\ o \in A \subset V$ – o- *продукція* і тому може статися, що $\varepsilon \in L(G)$ (нагадуємо, що $\varepsilon = o$ порожній ланцюжок).

Визначення 5.7. Формальна граматична структура з o- продукціями зветься o- *формальною граматичною структурою.* В протилежність до цього, якщо структура не містить у собі o- продукцій її звуть o- *вільною граматичною структурою.* Формальна мова зветься *порожньою* $L(G) = \{\varepsilon\}$, в структурі $C(G)$, якщо в цій структурі не виводиться ні один правильний ланцюжок або виводиться тільки ε- ланцюжок у випадку o- формальної граматичної структури.

Так, наприклад, формальна граматична структура з аксіоматикою (5.4) є o- вільною і вона породжує формальну мову L, за для якої $\varepsilon \notin L$. В подальшому «по замовченню» розглядаються формальні структури граматик, в яких породжуються тільки не порожні мови. Таким чином, ланцюжки формальної мови породжуються в заданій структурі за аксіоматикою згідно правильних виведень, причому кожен вивід, незалежно від того однозначний він чи ні, задає структуру виводу ланцюжка цієї мови. Вважається, що структура $C(G)$ *однозначна* (*частково однозначна*), якщо всі ланцюжки мови $L(G)$ мають

однозначні виводи. З попереднього прикладу зрозуміло, що в структурі з аксіоматикою (5.4) формальна мова має вигляд $L(G)=\{ab^k : k\in\mathbb{N}\}$ і її породжуюча структура $C(G)$ однозначна. Так при $k=n$ для ланцюжка ab^n однозначна структура виводу наступна

$$W(ab^n)=(\sigma, a\alpha, ab\alpha, abb\alpha, \ldots, ab^{n-1}\alpha, ab^n) \qquad (5.5)$$

і вивід супроводжується застосуванням спочатку аксіоми пріоритетного виводу, потім застосовується $n-1$ раз продукція виводу і аксіома виводу. Тому довжина виводу ланцюжка (5.5) – $\left|W(ab^n)\right|=n+1$. Прикладом неоднозначної формальної структури граматики $C(G_1)$ є структура з аксіоматикою

$$\Lambda_1=\begin{cases} \textit{аксіоми виводу}: \\ \left.\begin{array}{l}\alpha\to b\,|\,\alpha=b\\ \gamma\to b\,|\,\gamma=b\end{array}\right\}, \\ \sigma\to a\alpha-\textit{аксіома початку}, \\ \alpha\to\gamma, \\ \gamma\to b\gamma, \\ \alpha\to b\alpha. \end{cases} \qquad (5.6)$$

за якою мова $\{ab^k : k\in\mathbb{N}\}$ може бути породжена за допомогою різних виводів. Ланцюжок $l=abb$ можливо вивести, як $W(l)=(\sigma, a\alpha, ab\alpha, abb)$ або як $W_1(l)=(\sigma, a\alpha, a\gamma, ab\gamma, abb)$.

Виходячи з того, що структура ланцюжка l_l за його виводом $W_i(l)$ неоднозначна, введемо характеристику структури виводу – *складність виводу ланцюжка*.

Визначення 5.8. Складністю виводу ланцюжка l в структурі граматики $C(G)$ назвемо число $k(l)=\min\limits_i\{|W_i(l)|\}$.

Так для ланцюжка $l=abb$ породженого конструктивною структурою з аксіоматикою (5.6) його складність $k(l)=3$.

В програмуванні і інших пов'язаних з обробкою інформації ситуаціях важливо мати справу з однозначними конструктивними структурами, наприклад, при розробці програмних засобів на неоднозначній граматичній структурі можуть виникати проблеми виводу для правильного відтворити алгоритмічну схему розв'язку зада-

чі. Питання однозначності структур є основним питанням при розробці і побудові граматичних структур для заданих предметних областей і загальної теорії формальних структур та граматик.

З кожним виводом W довільного ланцюжка l в структурі $C(G)$ пов'язана *схема виводу* цього ланцюжка W_c. Так для структури з аксіоматикою (5.4), яку перепишемо з позначеннями продукцій:

$$\Lambda = \begin{cases} p_1 : \alpha \to b \mid \alpha = b - \textit{аксіома виводу}, \\ p_2 : \sigma \to a\alpha - \textit{аксіома приоритетного виводу}, \\ p_3 : \alpha \to b\alpha. \end{cases}$$

вивід $W(abb) = (\sigma, a\alpha, ab\alpha, abb)$ відповідає схемі виводу цього ж ланцюжка $W_c(abb) = (\sigma \xrightarrow{p_2} a\alpha \xrightarrow{p_3} ab\alpha \xrightarrow{p_1} abb)$.

Твердження 5.2. За однозначними схемами виводів ланцюжків $W_c(l)$ формальної мови $l \in L(G)$ однозначно відтворюється формальна структура граматики $C(G)$.

Таким чином, якщо прийняти за структуру виводу ланцюжка його схему виводу , тоді між частково однозначною формальною структурою граматики і структурою виводів всіх ланцюжків (не порожніх) її мови існує часткове взаємно однозначне відношення. Крім того для формальної мови $L(G)$ граматичної структури $C(G)$ має місце ланцюг по включенню.

Твердження 5.3. $L(G) \subset BWC(G) \subset WC(G) \subseteq \mathbb{F}(V)$.

5.3.3. Утворюючі граматичні структури

З твердження 5.2 виникає досить цікаве питання повноти утворюючих структур (підст*руктур)* формальної структури. Дамо пояснення цього поняття.

Розглянемо довільну граматичну структуру $C(G) = \langle V, \{\to^2\}, \Lambda \rangle$. Нехай V_1 *підсловник* словника V, а Λ_1 *підаксіоматика* аксіоматики Λ Λ, тоді з одного боку, за визначенням 4.11, $C_1 = \langle V_1, \{\to^2\}, \Lambda_1 \rangle$ є підструктура структури $C(G)$. Підаксіоматика Λ_1 може містити в собі деякі аксіоми пріоритету та аксіоми виводу, тоді підструктура C_1 може породжувати деякі ланцюжки мови $L(G)$, таку підструктуру назвемо

– породжуючою підструктурою структури $C(G)$ і позначатимемо її так C_1^*. Тому з другого боку, породжуюча підструктура C_1^* задає структуру (через спосіб виводу або схему виводу) деякого ланцюжка $l \in L(G)$.

Визначення 5.9. Об'єднанням двох конструктивних підструктур $C_1 = \langle V_1, \{\rightarrow^2\}, \Lambda_1 \rangle$ і $C_1 = \langle V_2, \{\rightarrow^2\}, \Lambda_2 \rangle$ називається структура $C_1 \bigcup C_2 = \langle V_1 \bigcup V_2, \{\rightarrow^2\}, \Lambda_1 \bigcup \Lambda_2 \rangle$, а відповідно їх перетином структура $C_1 \bigcap C_2 = \langle V_1 \bigcap V_2, \{\rightarrow^2\}, \Lambda_1 \bigcap \Lambda_2 \rangle$.

Твердження 5.4. Нехай сімейство $\{C_\alpha; \alpha \in I\}$ сукупність підструктур C_α структури $C(G)$. Тоді їх об'єднання і перетин утворюють підструктури формальної граматичної структури $C(G)$.

Визначення 5.10. Сукупність конструктивних підструктур $M_C = \left\{ C_\alpha;\ \alpha \in J,\ \bigcup_\alpha C_\alpha = C(G) \right\}$ називається *системою утворюючих підструктур* (M_C *– системою*) формальної структури $C(G)$.

Очевидно, що за заданою M_C- системою завжди можливо відтворити граматичну структуру і відповідну їй мову, тому і виникає проблема (*проблема повноти*): знайти такі критерії (умови), за якими вибрана система підструктур структури $C(G)$ була б M_C- системою утворюючих структур. В подальшому буде наведена *схема алгоритму* відтворення конструктивної структури.

5.3.4. Завдання і вправи

1. З'ясувати, чи утворюють породжуючі граматики наступні формальні упорядковані четвірки об'єктів:

а) $\langle \{a,b\}, \{\sigma,\gamma\}, \sigma, \{\sigma \rightarrow ab, \sigma \rightarrow a\sigma\gamma, \sigma\gamma \rightarrow b, b \rightarrow a\sigma\} \rangle$;

б) $\langle \{a,b\}, \{\sigma\}, \sigma, \{\sigma \rightarrow a\sigma ba\sigma, \sigma \rightarrow ab, a\sigma \rightarrow b\} \rangle$;

в) $\langle \{b,c\}, \{a,\sigma\}, \{\sigma \rightarrow a, a \rightarrow b, a \rightarrow ca\}, \sigma \rangle$;

г) $\langle \{b,c\}, \{\sigma,\alpha\}, \alpha, \{\alpha \rightarrow \sigma, \sigma \rightarrow b, \sigma \rightarrow c\sigma c\} \rangle$;

д) $\left\langle \begin{matrix} \{a,b,c\},\{\alpha,\beta,\sigma\},\{\alpha,\sigma\}, \\ \{\alpha \to a\sigma b, \sigma \to a\alpha c, \alpha \to a\beta b, \sigma \to a\beta c, \beta \to a\beta, \beta \to b \mid c\} \end{matrix} \right\rangle$

2. Відтворити формальні структури для породжуючих граматик завдання 1.

3. Дано термінальний алфавіт $\{a,b\}$ і алфавітний словник $\{a,b,\delta,\sigma\}$ з'ясувати можливість побудови породжуючих граматик за схемами:

а) $P = \{\delta \to b, \sigma \to a\delta b\}$;

б) $P = \{b \to a, \delta \to \sigma b, \sigma \to a\delta\}$;

в) $P = \{\delta \to b\delta a\sigma, a\sigma \to ab, \delta\sigma \to \varepsilon\}$;

г) $P = \{\delta \to b\delta\sigma, \sigma \to a, \delta\sigma \to a\sigma\}$;

д) $P = \{\delta \to b, \sigma \to a\sigma, \sigma \to \delta\delta\}$.

4. Побудувати класи безпосередньо виведених $BWC(G)$ і виведених $WC(G)$ у граматиці G ланцюжків якщо

$$G = \left\langle \{a,d,e\},\{\alpha,\beta,\sigma\},\sigma,\{\sigma \to a\beta, \beta \to \alpha d\sigma\beta, \alpha d \to ed, \beta \to ad\} \right\rangle:$$

навести класифікацію виведень множини $WC(G)$.

5. Задати формальні структури породжуючих граматик задачі 2.

6. Побудувати клас $WC(G)$ формальної структури $C(G)$, якщо

$$G = \left\langle \{a,b,c\},\{\alpha,\beta,\sigma,\gamma\},\alpha,\{\alpha \to a\beta, \beta \to \sigma\alpha\gamma, \beta \to \sigma b, \gamma \to b\sigma, \sigma \to c\} \right\rangle$$

і дати класифікацію його виведень.

7. Виписати ланцюжки, які породжуються в формальній структурі з завданою аксіоматикою:

$$\Lambda = \begin{cases} \left.\begin{matrix} \sigma \to b \mid \sigma = b, \\ \sigma \to c \mid \sigma = c; \end{matrix}\right\} - \textit{аксіоми виводу}; \\ \left.\begin{matrix} \sigma \to a\sigma a, \\ \sigma \to b \mid c; \end{matrix}\right\} - \textit{аксіоми приоритетного виводу}; \end{cases}$$

де введено скорочений запис продукцій $\sigma \to b \mid c$ замість двох – $\sigma \to b$, $\sigma \to c$. Який вигляд має формальна мова?

8. Для структури формальної граматики:

$$C(G)=\left\langle \begin{array}{l} \{a\},\{\alpha,\beta,\delta\},\{\alpha,\beta\}, \\ \left\{ \begin{array}{l} \left.\begin{array}{l}\alpha\to\alpha^k, k=1,2,\ldots,n;\\ \beta\to\delta^j,\ j=1,2,\ldots,m;\end{array}\right\} - \textit{аксіоми приоритетного виводу};\\ \delta\to\beta;\\ \delta\to\varepsilon\beta;\\ \delta\to\varepsilon - \textit{аксіома виводу};\end{array}\right\} \end{array}\right\rangle$$

довести, що формальна мова порожня $L(G)=\{\varepsilon\}$.

9. За структурою виводів ланцюжків формальних мов задачі 5 відтворити їх формальні структури.

10. Довести твердження 1.

11. Довести твердження 2.

12. Довести твердження 3.

13. Довести твердження 4.

14. Для формальної мови, яка породжена в структурі задачі 7 побудувати неоднозначну формальну структуру $C(G_1)$.

5.4. Загальні властивості граматичних структур

5.4.1. Нормальні граматичні структури

Раніше встановлено, що кожна формальна структура граматики породжує тільки одну певну формальну мову. Причому формальна мова може бути порожньою (дивись задачу 8, п. 5.3.4) або записана складним виразом (дивись, наприклад, мову задачі 7, п. 5.3.4). Тому між формальною структурою і мовою, яка в ній породжується існує складне відношення φ, причому

Твердження 5.5. Відношення φ не є взаємно однозначним (гомоморфним).

▷ Дійсно, нехай мова L виведена в структурі $C(G)$ і відношення між ними є φ . Розширимо не термінальний алфавіт N структури $C(G)$ новим символом $\xi\notin N$ в результаті чого отримаємо $N_1=N\bigcup\{\xi\}$. Розширимо також аксіоматику Λ структури $C(G)$ $C(G)$ продукцією $(\sigma\to\xi)\notin\Lambda,$ де $\sigma\in N$. Таким чином отримаємо іншу

структуру $C(G_1)$. Нескладно бачити, що мова L виводиться і в структурі $C(G_1)$ тобто відношення φ між структурою $C(G)$ і мово L зостається тим же – однозначним. Але його обернене φ^{-1} не є однозначним відношенням. $\triangleleft$

За твердженням 5.5 ми маємо можливість конструктивно побудувати скільки завгодно структур граматик, в яких виводиться одна і таж мова. Тому доцільно навести певний порядок на множині формальних граматичних структур CG. Як відомо з першого розділу, це можливо зробити за допомогою введення відношень еквівалентності.

Визначення 5.11. Дві формальні однотипові структури граматик $C_1, C_2 \in CG$ називаються слабко еквівалентними (еквівалентними), якщо вони породжують одну і ту ж формальну мову.

Введене відношення еквівалентності дозволяє за теоремою про розбиття (теорема 2.2) виділити у множині CG класи еквівалентних однотипових структур по відношенню до породжених ними мов. З'ясуємо деякі особливості будови цих класів. За для цього введемо поняття нормальної конструктивної граматичної структури.

Визначення 5.12. Структуру, в якій будь яка продукція або аксіома аксіоматики Λ має конструкцію

$$x \to y, \ x \in \mathbb{F}(N) \tag{5.7}$$

назвемо *нормальною формальною граматичною структурою* $C(G_h)$ і виведену в цій структурі мову *нормальною формальною мовою* L_h. Тобто аксіоматика структури $C(G_h)$ наділена властивістю (5.7).

Теорема 5.1. В будь якому класі еквівалентності множини CG завжди існує нормальна граматична структура $C(G_h)$.

$\triangleright$ Доведення теореми спирається на прийом розширення не термінального алфавіту і аксіоматики структури CG. Розглянемо клас еквівалентності структур відносно мови L – KC_L, і нехай $C(G) = \langle A \bigcup N, \Sigma, \Lambda \rangle$ довільна структура граматики цього класу.

Введемо додатні не термінальні символи $\alpha_i \notin N$ і поставимо їх у взаємно однозначну відповідність до елементів a_i алфавіту A так, що $\alpha_i \leftrightarrow a_i$. Тоді отримаємо розширений алфавіт $N_h = N \bigcup \{\alpha_i : i = 1, \ldots \# A\}$. Розширимо також аксіоматику Λ замінивши в аксіоматиці Λ всі термінальні символи $a_i \in A$ відповідними символами α_i і додавши до неї аксіоми виводу $\alpha_i \to a_i$. Зрозуміло, що

таким чином конструктивно побудована структура є нормальною граматичною структурою $C(G_h)$, при цьому між продукціями аксіоматик структур $C(G)$ і $C(G_h)$ за виключенням аксіом виводу $\alpha_i \to a_i$, на основі відношення $\alpha_i \leftrightarrow a_i$, існує взаємно однозначне відображення ϕ.

Нехай $x = a_{j_1}, a_{j_2}, \ldots a_{j_k}$ довільний ланцюжок мови L виведений у структурі $C(G)$, тобто $\sigma \Rightarrow x$, покажемо, що цей ланцюжок також виводиться і у структурі $C(G_h)$. Припустимо, що виведенню $\sigma \Rightarrow x$ відповідає вивід $W(x) = (\sigma, y_1, y_2, \ldots, y_n = x)$ тоді за існуванням відображення $\phi = (\phi_1, \phi_2, \ldots, \phi_r)$, $\phi_i : y_i \to y_i^h$ виводові $W(x)$ відповідає виведення $\sigma \Rightarrow y_n^h$ в структурі $C(G_h)$. Застосувавши далі аксіоми виводу з структури $C(G_h)$ отримаємо виведення $y_n^h \Rightarrow x$, або за транзитивністю виведення маємо виведення $\sigma \Rightarrow x$ в структурі $C(G_h)$. В результаті знайдено, що $L \subseteq L_h$.

Припустимо тепер, що ланцюжок $x \in L_h$ і йому відповідає вивід у структурі $C(G_h)$ $W_h(x) = (\sigma, y_1^h, y_2^h, \ldots, y_m^h = x)$. Зрозуміло, що у цьому виводі застосовуються безпосередні виводи за продукціями і аксіомами виводу аксіоматики структури $C(G_h)$, які не мають у своїх лівих частинах термінальних символів. Перелаштуємо вивід W_h так, щоб спочатку застосовувалися аксіома пріоритету і продукції виводу, а на при кінець аксіоми виводу. В результаті отримаємо вивід з двома послідовними виведеннями $\sigma \Rightarrow y_n^h$, $n < m$ – за аксіомами пріоритету та продукціями виводу і $y_n^h \Rightarrow x$ – за аксіомами виводу. Виходячи з того, що існує обернене відображення ϕ^{-1}, $\phi_i^{-1} : y_i^h \to y_i$ виведенню $\sigma \Rightarrow y_n^h$ можливо поставити в однозначну відповідність виведення $\sigma \Rightarrow y_n$, яке відповідає виводу $W(x)$ ланцюжка x структури $C(G)$, з чого маємо $L_h \subseteq L$.

Таким чином приходимо до висновку, що $L_h = L$. $\triangleleft$

Для прикладу розглянемо вправу з класом еквівалентності KC_L формальної мови $L = \{a^n b^n c^n;\ n \in \mathbb{N}\}$.

Приклад 5.2. Побудуємо нормальну формальну граматичну структуру еквівалентну до структури з аксіоматикою:

$$\Lambda = \begin{cases} p_1, p_2 : \sigma \to a\sigma\beta\gamma \mid a\beta\gamma - \textit{аксіоми приоритету}, \\ p_3 : \gamma\beta \to \beta\gamma, \\ \textit{аксіоми виводу}: \\ p_4 : a\beta \to ab, \\ p_5 : b\beta \to bb, \\ p_6 : b\gamma \to bc, \\ p_7 : c\gamma \to cc. \end{cases} \tag{5.8}$$

▷ Переконаємося, що в структурі з аксіоматикою (5.8) виводиться мова $L = \{a^n b^n c^n;\ n \in \mathbb{N}\}$. Схему виводу ланцюжків мови L продемонструємо на прикладі виведення ланцюжка $x = a^3b^3c^3$

$$\begin{aligned} \sigma \xrightarrow{p_1} a\sigma\beta\gamma \xrightarrow{p_1} aa\sigma\beta\gamma\beta\gamma \xrightarrow{p_2} a^3\beta\gamma\beta\gamma\beta\gamma \xrightarrow{p_4} \\ a^3b\gamma\beta\gamma\beta\gamma \xrightarrow{p_3} a^3b\beta\gamma\gamma\beta\gamma \xrightarrow{p_5} a^3bb\gamma\gamma\beta\gamma \xrightarrow{p_3} \\ a^3bb\gamma\beta\gamma\gamma \xrightarrow{p_3} a^3bb\beta\gamma\gamma\gamma \xrightarrow{p_5} a^3b^3\gamma\gamma\gamma \xrightarrow{p_6} \\ a^3b^3c\gamma\gamma \xrightarrow{p_7, p_7} a^3b^3c^3. \end{aligned} \tag{5.9}$$

Якщо тепер ввести додатні не термінали α, δ, λ і відношення $\alpha \leftrightarrow a,\ \delta \leftrightarrow b,\ \lambda \leftrightarrow c$, за якими змінити продукції аксіоматики Λ, тоді отримаємо аксіоматику структури $C(G_h)$

$$\Lambda_h = \begin{cases} p_1, p_2 : \sigma \to \alpha\sigma\beta\gamma \mid \alpha\beta\gamma - \textit{аксіоми приоритету}, \\ p_3 : \gamma\beta \to \beta\gamma, \\ p_4 : \alpha\beta \to \alpha\delta, \\ p_5 : \delta\beta \to \delta\delta, \\ p_6 : \delta\gamma \to \delta\lambda, \\ p_7 : \lambda\gamma \to \lambda\lambda \\ \textit{аксіоми виводу}: \\ p_8 : \alpha \to a, \\ p_9 : \delta \to b, \\ p_{10} : \lambda \to c. \end{cases} \tag{5.10}$$

Очевидно, що мова $L=\{a^n b^n c^n;\ n\in\mathbb{N}\}$ також виводиться в структурі $C(G_h)$. Наприклад, ланцюжок $x=a^3b^3c^3$ може бути виведений за тією ж схемою (5.9), але в ній під продукціями p_i слід розуміти відповідні продукції аксіоматики Λ_h і до отриманого таким чином ланцюжка $\alpha^3\delta^3\lambda^3$ застосувати аксіоми виводу структури $C(G_h)$. Виходячи з цього можливо стверджувати, що наведені вище структури з аксіоматиками (5.8) і (5.10) – слабко еквівалентні. Зауважимо, що до цього ж класу еквівалентності також належить і структура з аксіоматикою

$$\Lambda=\begin{cases} p_1, p_2:\sigma\to\alpha\sigma\beta \mid \alpha\delta c - \textit{аксіоми приоритету},\\ p_3: c\beta\to\gamma c,\\ p_4:\gamma\to\beta,\\ p_5: b\beta\to b\delta c\\ \textit{аксіоми виводу}:\\ p_6:\delta\to b,\\ p_7:\alpha\to a. \end{cases} \qquad (5.11)$$

З доведеної теореми 5.1 слідує, що в класі еквівалентних структур KC_L існує не одна нормальна структура, котрі утворюють підклас еквівалентних нормальних формальних структур KC_L^h. Тому має місце

Твердження 5.6. $KC_L^h \subset KC_L$

Досі при виведенні формальних мов в конструктивних структурах класів еквівалентності ми не звертали увагу на особливості виведення. Очевидно, що у будь якому класі еквівалентності існують однозначні і неоднозначні структури.

Визначення 5.13. Частково однозначні однотипові структури C_1 і C_2 звуть *сильно еквівалентними* $C_1 \tilde{\sim} C_2$ (), якщо вони породжують одну і ту ж мову і при цьому схеми виводів однакових ланцюжків у структурах C_1 і C_2 однакові.

Прикладом сильної еквівалентності структур є структури з аксіоматиками:

$$\Lambda' = \begin{cases} \sigma \to a\sigma a \mid \tau\tau - \textit{аксіоми приорітету}, \\ \textit{аксіоми виводу}: \\ \sigma \to c, \\ aca \to b; \end{cases}$$

$$\Lambda'' = \begin{cases} \sigma \to a\sigma a - \textit{аксіома приорітету}, \\ \textit{аксіоми виводу}: \\ \sigma \to c, \\ aca \to b. \end{cases}$$

За допомогою введення на однозначних структурах поняття сильної еквівалентності можливо в класі еквівалентних структур KC_L виділити підклас сильно еквівалентних структур $\widetilde{KC_L}$, тобто $\widetilde{KC_L} \subset KC_L$. Не складно бачити, що в класі нормальних еквівалентних структур також можливо виділити підклас сильно еквівалентних структур $\widetilde{KC_L^h}$, таким чином за твердженням 5.5 маємо

Твердження 5.7. $\widetilde{KC_L^h} \subset KC_L^h \subset KC_L$.

Введене вище поняття частково однозначної формально граматичної структури за допомогою структур виводу ланцюжків або схем цих виводів є фактично слабко однозначним визначенням однозначності структур. Дійсно, розглянемо формальну структуру з наступною аксіоматикою:

$$\Lambda = \begin{cases} p_1, p_2 : \alpha \to \alpha\alpha \mid aa - \textit{аксіоми приорітету}, \\ p_2 : \alpha \to aa - \textit{аксіома виводу}. \end{cases} \qquad (5.12)$$

яка породжує мову $\{a^{2n}; n \in \mathbb{N}\}$.

Очевидно, що схема виводу ланцюжка a^6 буде неоднозначною на етапі застосування аксіоми пріоритету $\alpha \to \alpha\alpha$ і аксіоми виводу, при цьому неоднозначність у другому випадку схема виводу розпізнає, а у першому випадку вона не розпізнається виводом. Наприклад, при такій початковій частині схеми виводу $\alpha \xrightarrow{p_1} \alpha\alpha \xrightarrow{p_1} \alpha\alpha\alpha$ не зрозуміло до якого символу α у другому безпосередньому виводові застосовується продукція p_1. Для виправлення такого становища введемо поняття *способу виводу ланцюжка* $l - S_W(l)$ [16].

Розглянемо деяке виведення $x \Rightarrow y \equiv x = z_0 \mapsto z_1 \mapsto z_2 \mapsto \cdots z_k = y$ в завданій структурі, нехай його довільний безпосередній вивід $z_i \mapsto z_{i+1}$ має вигляд $u_i x_i v_i \xrightarrow{p_j} u_i y_i v_i$, де $u_i, x_i, v_i, y_i \in \mathbb{F}(V)$. Позначимо через q_i довжину підланцюжка u_i ланцюжка z_i і створимо пару $[q_i, p_j]$. Побудована таким чином пара завдає *спосіб безпосереднього виводу*. Зрозуміло, що між способом безпосереднього виводу $[q_i, p_j]$ і безпосереднім виводом $z_i \mapsto z_{i+1}$ не існує взаємно однозначного відношення. Це випливає хоча б з того, що безпосередньому виводові $\alpha\alpha \mapsto \alpha\alpha\alpha$ в структурі з аксіоматикою (5.12) відповідає два способи безпосереднього виводу: $[1, p_1]$ і $[2, p_1]$. Послідовність способів безпосереднього виводу $\left([q_0, p_r], [q_1, p_j], \cdots [q_{k-1}, p_s]\right)$, яка відповідає виведенню $x \Rightarrow y$ і називається *способом виводу ланцюжка* y.

Визначення 5.14. Формальна граматична структура $C(G)$ називається *однозначною*, якщо в ній кожен ланцюжок виводиться за допомогою одного і тільки одного свого способу виводу. Відповідно породжена в такій структурі множина називається *однозначною мовою.*

Очевидно, визначення 5.14 є більш коректним, ніж визначення однозначності граматичної структури через однозначність виводів, наведене у попередньому пункті. Прикладом однозначної формальної структури є структура з аксіоматикою:

$$\Lambda = \begin{cases} \sigma \to \alpha \mid \sigma\alpha \mid \sigma\beta - \textit{аксіоми приорітету}, \\ \textit{аксіоми виводу}: \\ \alpha \to a \mid b \mid \cdots \mid z, \\ \beta \to 0 \mid 1 \mid \cdots \mid 9; \end{cases}$$

яка породжує класичну однозначну мову ідентифікаторів, котрі застосовуються в мовах програмування для позначення імен: файлів, змінних, функцій, масивів і інше. В однозначній формальній структурі між будь яким безпосереднім виводом і його способом безпосереднього виводу існує взаємно однозначне відношення, отже існує взаємно однозначне відношення між схемою виводу мовного ланцюжка і способом виводу цього ланцюжка. Тому тепер можливо переформулювати визначення сильної еквівалентності 5.13 наступним чином:

Визначення 5.15. Однозначні однотипові структури C_1 і C_2 звуть сильно еквівалентними ($C_1 \tilde{\sim} C_2$), якщо вони породжують одну і ту ж

мову, і схеми виводів (способи виводів) однакових ланцюжків у структурах співпадають.

Це досить жорстке визначення 5.15 сильної еквівалентності приводе до деяких незручностей, наприклад, не всякий клас сильно еквівалентних структур має в собі нормальну граматичну структуру (вище наведена теорема 5.1 не справедлива за умови співпадання способів виводу). Таким чином сильно еквівалентні класи «бідніші», ніж еквівалентні класи за визначенням 5.10.

Нагадаємо, що вище розглянутий підхід до виводу ланцюжків мови через виведення за способом виводу виник з причини не розпізнання за схемою виводу місця в ланцюжкові, до якого застосовується рекурсивна продукція з повторенням $p: y \to yy\cdots y,\ y \in \mathbb{F}(V)$. Але продукції типу p формальної структури завжди можуть бути модифіковані так, щоб не змінити мову і встановити взаємно однозначне відношення між схемами виводів і способами виводів ланцюжків цієї формальної мови.

Твердження 5.8. Модифікація продукції $p: y \to yy,\ y \in \mathbb{F}(V)$ парою продукцій

$$\begin{cases} p_1 : y \to xy, \\ p_2 : x \to y; \end{cases} \text{ або } \begin{cases} p_1 : y \to yx, \\ p_2 : x \to y; \end{cases} \quad x \in \mathbb{F}(V),\ |x| \le |y|; \tag{5.13}$$

не змінює породжених ланцюжків і встановлює взаємно однозначне відношення між схемами виводів і способами виводів ланцюжків.

▷ Дійсно, ланцюжки, які виводиться в структурі з продукцією p є тими ж, що і в структурах з продукціями (5.13).

Розглянемо тепер довільний вивід ланцюжка в структурі з продукцією p, нехай це буде – (y, yy, yyy). Цьому виводові відповідає два способи виводів: 1) $([1, p],[1, p])$ і 2) $([1, p],[2, p])$. Нескладно перевірити, що будь якому виводові ланцюжка в структурі з групою продукцій (5.13) відповідає завжди один спосіб виводу, наприклад, виводу за першою групою продукцій (5.13) – (y, xy, yy, xyy, yyy) відповідає наступний спосіб виводу: $([1, p_1],[1, p_2],[1, p_1],[1, p_2])$. ◁

Спираючись на результати твердження 5.4, будемо в подальшому розглядати структури, в аксіоматиці яких будуть відсутні продукції типу $p: y \to yy\cdots y,\ y \in \mathbb{F}(V)$ Тоді визначення 5.8 та 5.11 і визначення 5.10 та 5.12 будуть еквівалентними за взаємно однозначністю схем виводів і способів виводів ланцюжків. Так для структури з аксіома-

тикою (5.12) еквівалентною буде структура з модифікованою аксіоматикою

$$\tilde{\Lambda} = \begin{cases} p_1, p_2 : \sigma \to \alpha\sigma \mid aa - \textit{аксіоми приорітету}, \\ p_2 : \sigma \to aa - \textit{аксіома виводу}, \\ p_3 : \alpha \to \sigma. \end{cases} \qquad (5.14)$$

Як було сказано вище, в програмуванні необхідно мати справу з однозначними конструктивними структурами. Взагалі то проблема розпізнання однозначності граматичних структур не розв'язана, тобто не вдається побудувати такий алгоритм, за яким би для будь якої структури з заданого класу можливо встановити її однозначність. Більш детальну визначеність відносно підкласів однозначних, o_o-вільних і інших формально граматичних структур можливо отримати після введення деяких додатних властивостей у аксіоматику формальних граматичних структур.

5.4.2. Класи Хомського

Американським лінгвістом Н. Хомським [13] виконана класифікація породжуючих формальних граматик і виділено чотири типи граматик: граматики нульового типу (граматики без обмежень на продукції), граматики першого типу або *контекстно залежні граматики* (*граматики безпосередніх складників*), граматики другого типу або *контекстно вільні граматики* (*безконтекстні*) і граматики третього типу – *автоматні граматики*. Спираючись на цю класифікацію введемо наступні типи породжуючих формальних граматичних структур:

1) *контекстно залежну структуру* (*безпосередніх складників*) $BC(G)$, продукції аксіоматики якої задовольняють наступним умовам

$$p_j : u_i\alpha_k v_i \to u_i y_m v_i,\ u_i, v_i, y_m \in \mathbb{F}(V);\ \alpha_k \in N; \qquad (5.15)$$

де підланцюжки u_i, v_i ланцюжків продукцій p_j є контекстним супроводженням;

2) *контекстно вільну структуру (безконтекстну)* $VC(G)$ *з продукціями аксіоматики*

$$p_j : \alpha_i \to y_k,\ \alpha_i \in N;\ y_k \in \mathbb{F}(V); \qquad (5.16)$$

3) автоматну структуру $AC(G)$, *продукції якої мають властивості:*

$$p_j : \alpha_i \to a_k \beta_n,\ \alpha_i, \beta_n \in N;\ a_k \in A; \text{ або } p_j : \alpha_i \to a_k. \qquad (5.17)$$

Формальні мови відповідно назвемо: *контекстно залежними* (*безпосередніх складників*) $BL(G)$, *контекстно вільними* (*безконтекстними*) $VL(G)$ і *автоматними* $AL(G)$.

Аналізуючи введені типи структур приходимо до висновку, що умови продукцій (5.16) є частковий випадок умов продукцій (5.15) $(u_i = v_i = \varepsilon)$, а властивості продукцій (5.17) є частковим випадком обмежень, накладених на продукції (5.16); тому можливо стверджувати, що між відповідними мовами існує ланцюг за включенням: $AL \subset VL \subset BL \subset \mathbb{F}(V)$.

Тепер, спираючись на результат зв'язаності за включенням мов AL, VL, BL зробимо висновок, що структура $AC(G)$ є підструктурою структури $VC(G)$, а в сою чергу структура $VC(G)$ є підструктурою структури $BC(G)$ по відношенню до відповідних формально граматичних мов.

Зауважимо, що з наведених чотирьох типів структур тільки для структур третього типу вище вказана проблема однозначності алгоритмічно розв'язана [3]. Тому в наступних розділах посібника будуть розглядатися в основному однозначні формальні граматичні структури.

5.4.3. Завдання і вправи

1. Довести, що визначення еквівалентності 11 є відношенням еквівалентності.
2. Довести, що формальні структури з аксіоматиками: $\Lambda_0, \Lambda_1, \Lambda_2$ – слабко еквівалентні; якщо

$$\Lambda_0 = \begin{cases} \sigma \to a\sigma a \mid \tau\tau \mid c - \textit{аксіоми приорітету}; \\ \textit{аксіоми виводу}: \\ \sigma \to c, \\ aca \to b; \end{cases}$$

$$\Lambda_1 = \begin{cases} \sigma \to a\sigma a \mid b \mid c - \textit{аксіоми приорітету}; \\ \textit{аксіоми виводу}: \\ \sigma \to c, \\ \sigma \to b; \end{cases}$$

$$\Lambda_2 = \begin{cases} \sigma \to a\sigma a \mid a^2\sigma a^2 \mid c - \textit{аксіоми приорітету}; \\ \textit{аксіоми виводу}: \\ \sigma \to c, \\ aca \to b. \end{cases}$$

3. Показати, що серед структур вправи 2 є сильно еквівалентні структури.
4. Для структур вправи 2 побудувати їм еквівалентні нормальні формальні граматичні структури.
5. Для структури з аксіоматикою Λ_0 вправи 2 побудувати сильно еквівалентні нормальні граматичні структури.
6. Побудувати нормальні граматичні структури слабко еквівалентні до структури з аксіоматикою (5.12).
7. Для формальної мови $\{a^k b^n c^m;\ k,n,m \in \mathbb{N}\}$ побудувати однозначну формальну граматичну структуру і її слабко еквівалентну нормальну структуру.
8. З'ясувати однозначність структур вправи 2.
9. Для ланцюжків $l = (a^i b^j c^k;\ i = 1,2,3;\ j = 2,3,4;\ k = 1,3,5)$ виведених в однозначної структури вправи 7 записати виводи, схеми виводів, способи виводів.
10. Для ланцюжків $l_i = (a^i b^i c^i;\ i = 2,3,4)$ побудувати граматичну структуру; відтворити схеми виводів, виводи, способи виводів.
11. За формальними мовами побудувати формальні граматичні структури:

а) $\{a^n b^m;\ n \in \mathbb{N},\ m \in \mathbb{N}_0\}$,

б) $\{a^n b^n;\ n \in \mathbb{N}\}$,

в) $\{\{a^n b^n\} \cup \{b^m a^m\};\ n, m \in \mathbb{N}\}$,

г) $\{a^n b^m c^n;\ n \in \mathbb{N},\ m \in \mathbb{N}_0\}$,

д) $\{a^n b^n a^n;\ n \in \mathbb{N}\}$,

е) $\{a^n b^n c^m;\ n \in \mathbb{N},\ m \in \mathbb{N}_0\}$,

ж) $\{\{a^n b^n c^m\} \cup \{a^m b^k c^k\};\ n \in \mathbb{N},\ m \in \mathbb{N}_0\}$,

з) $\{\{a^n b^n c^m\} \cup \{a^m b^n c^n\};\ n, m \in \mathbb{N}\}$,

і) $\{a^n b^n a^n b^m;\ n, m \in \mathbb{N}\}$,

и) $\{((x)^n (y)^m)^k;\ x = a + b,\ y = b + a,\ n, m, k \in \mathbb{N}\}$.

12. Визначити типи мов завдання 11.
13. Для мов завдання 11 побудувати нормальні формальні структури.
14. Для кожної з мов завдання 11 виписати множини ланцюжки, довжина виводів яких не більша за п'ятнадцять.
15. Записати способи виводів множин ланцюжків отриманих у вправі 13.
16. За множиною продукцій:

$$\begin{cases} \sigma \to \gamma, \\ \gamma \to \alpha_1 \mid \delta \mid \beta_1 \mid \beta_2,\ \alpha_1 \to \alpha, \\ \delta \to a\delta \mid a\alpha_1, \\ \beta_1 \to b\beta_1 \mid b\delta \mid b\alpha_1, \\ \beta_2 \to a\beta_2 \mid a\beta_1 \mid a\alpha_2,\ \alpha_2 \to \alpha,\ \alpha \to o; \end{cases}$$

граматики G з початком виводу σ побудувати нормальну формальну структуру та породжену нею мову. Визначити тип структури і її однозначність.
17. Якими способами можливо вивести ланцюжок (a^{10}) у граматичній структурі з аксіоматикою вправи 16?
18. Що необхідно зробити у аксіоматиці вправи 16 щоб спосіб виводу цього ланцюжка був однозначним?

5.5. Питання повноти системи утворюючих підструктур формальної граматичної структури

Розглянемо методику, за допомогою якої можливо побудувати повну систему утворюючих підструктур формальної граматичної структури. При цьому ми будемо спиратися на деякі алгебраїчні результати повноти утворюючих підалгебр (дивись, наприклад [16]). Для цього розглянемо більш детально деякі властивості підструктур граматичної структури.

5.5.1. Властивості підструктур

Будь яка підструктура C_1 граматичної структури $C(G)$ може бути *повною* або *не повною*, породжуючою C_1^* або не породжуючою. Серед сукупності підструктур $\{C_i\}$ граматичної структури $C(G)$ існують такі підструктури C_j, аксіоматика яких Λ_j повністю відтворює їх формальну структуру, тобто $C_j = \langle V_j, \Sigma, \Lambda_j \rangle$. Назвемо такі підструктури *повними підструктурами* $\overline{C}_j$, а відповідні їм аксіоматики *відтворюючими аксіоматиками* $\overline{\Lambda}_j$. Так для структури $C(G)$ з аксіоматикою

$$\Lambda = \begin{cases} p_1 : \xi \to \alpha\alpha\beta, \text{ аксіома приорітету}; \\ p_2 : \alpha\beta \to \delta\beta\beta; \\ p_3 : a\beta\beta \to ab\beta; \\ p_4 : \beta \to \gamma; \\ \text{аксіоми виводу}: \\ p_5 : \alpha \to a; \\ p_6 : \delta \to a; \\ p_7 : \gamma \to c; \end{cases} \tag{5.18}$$

повні граматичні підструктури

$$\overline{C}_1^* = \langle \{a, c, \alpha, \beta, \gamma, \xi\}, \Sigma, \{p_1, p_4, p_5, p_7\} \rangle,$$

$$\bar{C}_2 = \left\langle \{a,b,c,\alpha,\beta,\delta,\gamma\}, \Sigma, \{p_2, p_3, p_4, p_6, p_7\} \right\rangle,$$

неповні формальні підструктури

$$C_3 = \left\langle \{a,c,\alpha,\beta,\xi\}, \Sigma, \{p_1, p_4, p_5, p_6\} \right\rangle,$$

$$C_4^* = \left\langle \{a,b,c,\alpha,\beta,\delta,\gamma,\xi\}, \Sigma, \{p_1, p_4, p_5, p_7\} \right\rangle.$$

Визначення 5.16. Підструктура C_1 є *порожньою підструктурою* формальної граматичної структури $C(G)$, якщо вона породжує тільки порожню мову $L = \varnothing$, тобто 1) $V_1 = \{o\} = \varnothing$ або 2) $\Lambda_1 = \{x_i \to o;\ i \in J\}$, або 3) $V_1 = \{o\}$ і $\Lambda_1 = \{x_i \to o;\ i \in J\}$ або 4) аксіоматика Λ_1 не має усіх спільних символів зі словником V_1.

Наприклад, для формальної структури $C(G)$ з аксіоматикою (5.18) наступна підструктура $C_1 = \left\langle \{b,\alpha,\beta,\gamma,\xi\}, \Sigma, \{p_1, p_4, p_5, p_7\} \right\rangle$ є порожньою.

Зауваження 5.1. Наведене визначення порожньої підструктури за умовами 1) – 4) еквівалентні згідно визначенню 5.14.

У множині усіх підструктур формальної структури містяться також *ізольовані* підструктури, які визначаються так

Визначення 5.17. Підструктура C_1 називається ізольованою відносно підструктури C_2 у формальній структурі $C(G)$, якщо $C_1 \subset C_2$ і серед виведених ланцюжків $\{l_i \mid l_i \in \mathbb{F}(V_2)\}$ у структурі C_2 знайдеться хоча б один ланцюжок l_j такий, що $l_j \notin \{l_k \mid l_k \in \mathbb{F}(V_1)\}$. Якщо підструктура C_1 ізольована відносно структури $C(G)$, тоді підструктура C_1 зветься *ізольованою* у формальній структурі $C(G)$ і позначимо це так $C_1 \multimap C(G)$.

Для наведеної вище структури з аксіоматикою (5.18) ізольованою є підструктура $\bar{C}_1^* \multimap C(G)$, яка породжує ланцюжок $aac \in L(G) \subset \mathbb{F}(V)$.

Наведемо деякі властивості відношення $\multimap$. Будемо вважати, що порожня структура $\varnothing$ є ізольованою до будь якої структури. Якщо прийняти для будь якої підструктури $C_1 \subseteq C(G)$, що $C_1 \multimap C_1$, тоді відношення $\multimap$ є відношенням часткового порядку на множині усіх підструктур формальної структури $C(G)$, бо виконуються умови антисиметрії $\left(C_1 \multimap C_2,\ C_2 \multimap C_1;\ C_1 = C_2\right)$ та транзитивності

$(C_1 \multimap C_2,\ C_2 \multimap C_3;\ C_1 \multimap C_3)$. За твердженням 5.4 та визначенням 5.16 для сімейства підструктур $\{C_i\}$ структури $C(G)$ таких, що $C_i \multimap C_1$, маємо

Наслідок 5.1. $\bigcap_i C_i \multimap C_1$ у формальній структурі $C(G)$.

Ізольовані підструктури відносно формальної структури $C(G)$ можуть бути породжувальними і повними підструктурами.

Перейдемо тепер до розгляду систем утворюючих підструктур граматичної структури (див. визначення 5.10). Виходячи з того, що словник V і аксіоматика Λ формальної граматичної структури $C(G)$ є скінченими приходимо до висновку, що система утворюючих підструктур $\{C_i\}$ також є скінченою множиною.

Якщо система утворюючих підструктур M_C структури $C(G)$ складається з двох підструктур C_1 і C_2, таких що $C_1 \bigcup C_2 = C(G)$ і $C_1 \bigcap C_2 = \varnothing$, то підструктура C_2 є *доповненням* підструктури C_1 до формальної структури $C(G)$. Позначається доповнення структури C_1 структурою C_2, як $\widehat{C}_2$.

Лема 5.1. Нехай підструктура $\widehat{C}_2$ є доповненням підструктури C_1 до формальної структури $C(G)$, тоді множину усіх підструктур $\{C_i\}$ структури $C(G)$ можливо розбити на три класи підструктур:

$K_1 = \{C_j \mid C_j \subseteq C_1\}$, $K_2 = \{C_j \mid C_j \subseteq \widehat{C}_2\}$ і

$K_3 = \{C_j \mid C_j \bigcap C_1 \in K_1,\ C_j \bigcap \widehat{C}_2 \in K_2\}$.

Наслідок 5.2. Лема 5.1 має місце і в тому випадку, коли доповнення $\widehat{C}_2 = \bigcup_k C_k$, де $C_k \not\subset C_1$ підструктури формальної структури $C(G)$, а також коли підструктура є C_1 ізольованою, тобто $C_1 \multimap C(G)$.

Результат леми розбиття на класи є корисним при визначені будови множини підструктур формальної структури, зокрема будови її системи утворюючих підструктур.

Покажемо, що у довільній системі утворюючих підструктур структури $C(G)$ містяться повна $\bar{M}_C$ та породжуюча M_C^* утворюючі підсистеми.

Теорема 5.2. У будь якій системі утворюючих підструктур $M_C = \left\{C_i;\ i \in I,\ \bigcup_i C_i = C(G)\right\}$ граматичної структури $C(G)$ завжди

можливо виділити множину $M_C^* \subseteq M_C$ утворюючих породжуючих підструктур таку, що $M_C^* = \left\{ C_i^*;\, i \in I,\, \bigcup_i C_i^* = C(G) \right\}$.

▹ Для доведення теореми достатньо показати, що $M_C^* \subset M_C$. Припустимо протилежне, тобто $M_C^* \not\subset M_C$. За умовою теореми система M_C є утворюючою структури $C(G)$, яка породжує формальну мову $L(G)$. Звідси маємо, що серед підструктур системи M_C існують підструктури C_j^*, в яких виводяться ланцюжки $l \in L(G)$. Отже маємо таку залежність для підструктур $C_j^* \in M_C$, що призводе до протиріччя відносно припущення. ◃

Теорема 5.3. У всякій системі утворюючих підструктур M_C граматичної структури $C(G)$ міститься система утворюючих повних підструктур $\overline{M}_C = \left\{ \overline{C}_j;\; j \in J,\; \bigcup_j \overline{C}_j = C(G) \right\}$.

▹ Розглянемо довільну підструктуру $C_i \in M_C$. Якщо ця структура не є o- вільною, тобто на її словникові V_i за аксіоматикою Λ_i Λ_i можливо вивести тільки порожній ланцюжок тоді за визначенням 5.15 при умовах 1), 3) і 4) підструктура C_i – порожня. За зауваженням 5.1 замінимо структуру C_i еквівалентною структурою з умовою 2) так, що аксіоматика $\overline{\Lambda}_j$ буде складатися тільки з продукцій виду 2), після заміни правих частин продукцій порожнім символом o і словник $\overline{V}_j$ створимо з різних символів аксіоматики $\overline{\Lambda}_j$. Таким чином у цьому випадку маємо $\overline{C}_j \in \overline{M}_C$.

Нехай тепер структура C_i не порожня, тоді приймемо аксіоматику Λ_i за аксіоматику $\overline{\Lambda}_j$, при цьому можливі випадки:

1) аксіоматика Λ_i повністю відтворює структуру C_i, тобто $V_i = \overline{V}_j$;

2) аксіоматика Λ_i не повністю відтворює структуру C_i, але $\overline{V}_j \subset V_i$.

З чого по сукупності випадків 1) і 2) слідує включення, $\overline{C}_j \subset C_i$.

Якщо ж для аксіоматики $\Lambda_i = \overline{\Lambda}_i$ маємо $V_i \subset \overline{V}_i$, то розбиваючи аксіоматику $\Lambda_i = \overline{\Lambda}_j \bigcup \Lambda_{i-j}$ так, щоб $\overline{V}_j \subseteq V_i$ отримаємо і в цьому випадку включення $\overline{C}_j \subset C_i$. На цьому завершується доведення теореми. ◁

Зауваження 5.2. За результатом теореми 5.2 маємо для систем утворюючих підструктур наступне включення $\overline{M}_C^* \subseteq \overline{M}_C$ і за теоремою 5.3 – ланцюг по включенню $\overline{M}_C^* \subseteq \overline{M}_C \subseteq M_C$.

З'ясуємо тепер питання критеріїв, за якими можливо встановити існування систем утворюючих підструктур формальної граматичної структури і визначимо ефективні критерії, за якими можливо відтворити граматичну структуру за структурами утворюючих ланцюжків заданої формальної мови.

Для розв'язку проблема існування критеріїв про знаходження систем утворюючих підструктур скористуємося алгебраїчним підходом, який спирається на застосування максимальних підалгебр універсальних алгебр [17].

Нехай $C(G)$ довільна граматична структура (5.3), тоді підструктура C_m структури $C(G)$ називається *максимальною підструктурою* $C_m \subset C(G)$, якщо не існує такої підструктури $C_1 \subset C(G)$, за для якої мало б місце власне включення $C_m \subset C_1$. Позначимо через p_i довільну продукцію аксіоматики $\Lambda\wedge$ структури $C(G)$. Тепер, як нескладно бачити, підструктура C_m буде максимальною відносно структури $C(G)$ тоді і тільки тоді, коли для будь якого елементу $v \in C(G) \setminus C_m$ такого, що $v \in V \bigcup \{p_i, i \in I\}$ має місце $C_m \bigcup \{v\} = C(G)$. Тут під різницею $C(G) \setminus C_m$ розуміється підструктура $\langle V \setminus V_m, \Sigma, \Lambda \rangle$ або $\langle V, \Sigma, \Lambda \setminus \Lambda_m \rangle$, або $\langle V \setminus V_m, \Sigma, \Lambda \setminus \Lambda_m \rangle$ граматичної структури $C(G)$.

Граматична структура $C(G)$ має скінчену кількість максимальних підструктур. Позначимо через M множину усіх підструктур максимальних відносно структури $C(G)$. Для подальшого необхідна наступна лема розширення будь якої підструктури граматичної структури $C(G) = \langle V, \Sigma, \Lambda \rangle$.

Лема 5.2. Будь-яку підструктуру $C_1 \subset C(G)$ можливо розширити до максимальної підструктури $C_m \in M$ структури $C(G)$.

▷ За ствердженням леми маємо, що для довільної підструктури C_1 структури $C(G)$ у множині M існує така підструктура C_m, що можливе тільки таке включення $C_1 \subseteq C_m$, бо у протилежному випадку ви-

конується включення $C_1 \supset C_m$ і підструктура C_1 не є власною підструктурою структури $C(G)$. Припустимо, що для підструктури C_1 у множині M не існує підструктури $C_m \supset C_1$. Тоді приєднуючи до підструктури C_1 усі елементи $v \in V \bigcup \{p_i, i \in I\}$, яких нема у цій підструктурі крім одного $v^* \notin C_1$ за скінчену кількість кроків отримаємо максимальну підструктуру $C_{1,m}$ відносно структури $C(G)$, що призводе до протиріччя з припущенням. Таким чином будь яку власну підструктуру граматичної структури завжди конструктивно можливо розширити до максимальної підструктури. $\triangleleft$

Наприклад, для граматичної структури з аксіоматикою (5.18) порожню підструктуру $C_1 = \langle \{b, \alpha, \beta, \gamma, \xi\}, \Sigma, \{p_1, p_4, p_5, p_7\} \rangle$ можливо розширити неоднорідними об'єктами $\{a, c, \delta, p_2, p_3\}$ (доповнюючи словник $V_1 \bigcup \{a, c, \delta\}$ і аксіоматику $\Lambda_1 \bigcup \{p_2, p_3\}$) до максимальної підструктури $C_{1,m} = C(G) \setminus \{p_6\} \in M$.

5.5.2. Утворюючі підструктури і критерій повноти

Перейдемо до розгляду критерію, за яким визначається, що система підструктур граматичної структури є утворюючою системою. За певною аналогією структур з алгебрами назвемо його *критерієм Поста*, як це зроблено в алгебрах [18].

Теорема 5.4. Для того, щоб система підструктур $M_C = \{C_i; i \in I\}$ граматичної структури $C(G)$ була утворюючою системою необхідно і достатньо, щоб для будь якої підструктури $C_m \in M$ у системі M_C знайшовся хоча б один елемент $v \in \left\{V_i \bigcup \{p_{i,j}; j \in J\}; i \in I\right\}$ такий, що $v \notin C_m$.

$\triangleright$ За необхідністю система M_C – утворююча відносно структури $C(G)$, тобто $\bigcup_{i \in I} C_i = C(G)$ для максимальної підструктури $C_m \in M$, за її визначенням існує такий елемент $v \notin C_m$, що $v \in C(G) \setminus C_m$, тому в системі M_C знайдеться хоча б одна підструктура C_j для якої $v \in C_j$.

При доведенні достатності розглянемо таку систему M_C, в котрій для будь якої максимальної підструктури $C_m \in M$ структури $C(G)$ існує хо-

ча б один елемент $v \in C_i \in M_C$ такий, що $v \notin C_m$. Доведемо, що система M_C є утворюючою, тобто виконується умова $\bigcup_{i \in I} C_i = C(G)$.

Припустимо, що система M_C не є системою утворюючих підструктур – $\bigcup_{i \in I} C_i \neq C(G)$, тоді користуючись результатами леми 5.2, будь яку підструктуру $C_i \in M_C$ розширимо до максимальної підструктури $C_{i,m} \in M$ структури $C(G)$, з чого маємо включення $C_i \subset C_{i,m}$. Але за умовою у підструктурі C_i існує такий елемент v_i, для якого $v_i \notin C_{i,m}$, що призводе до порушення включення $C_i \subset C_{i,m}$. Таким чином наше припущення про те, що сукупність підструктур M_C не є системою утворюючих підструктур породжуючої граматичної структури $C(G)$ хибне і теорема доведена. $\triangleleft$

Зауваження 5.3. Результати теореми 5.4 мають місце і для систем породжуючих підструктур M_C^* і повних систем підструктур $\bar{M}_C$ формальної граматичної структури $C(G)$.

Система утворюючих підструктур називається *повною системою* M_C^m у тому розумінні, що вона повністю відтворює формальну структуру C і в ній нема зайвих підструктур, які не впливають на відтворення структури $C(G)$.

Отже за результатами леми 5.1 та теореми Теорема 5.4 можливо запропонувати наступну схему побудови системи утворюючих підструктур формальної структури і дослідити будову цієї утворюючої системи підструктур:

1) побудувати множину максимальних підструктур M;
2) за елементами, які не входять до максимальних підструктур побудувати систему утворюючих підструктур M_C, що містять у собі ці відсутні елементи;
3) на системі утворюючих підструктур формальної системи побудувати структурний граф залежності підструктур;
4) виділити у системі утворюючих підструктур ізольовану підструктуру і доповнену до неї підструктуру, відносно яких за лемою 5.1 побудувати три класи K_i; $i = 1,2,3$;
5) з класів K_1 і K_2 виділити незалежні підструктури, тобто такі $C_i^1, C_i^2 \in K_i$ для яких $C_i^1 \bigcap C_i^2 \notin K_i$ і $C_i^1 \not\subset C_i^2$ або $C_i^2 \not\subset C_i^1$;

6) на об'єднані незалежних структур класів K_1 і K_2 побудувати повну систему утворюючих підструктур M_C^m формальної структури.

Застосуємо наведену схему до формальної структури з аксіоматикою

$$\Lambda = \begin{cases} p_1 : \sigma \to a\alpha, \text{ аксіома приорітету}; \\ p_2 : \alpha \to a\alpha; \\ p_3 : \alpha \to b\beta; \\ p_4 : \beta \to b\beta; \\ p_5 : \beta \to c\gamma; \\ p_6 : \gamma \to c\gamma; \\ \text{аксіоми виводу}: \\ p_7 : \gamma \to c; \\ p_8 : \beta \to c. \end{cases} \tag{5.19}$$

та мовою $L = \{a^k b^n c^m;\ k, n, m \in \mathbb{N}\}$.

Побудуємо породжуючу систему $\bar{M}_C^*$.

Для неї і максимальної множини M достатньо скористуватися відповідними аксіоматиками. Множиною аксіоматик сукупності M є $\{\Lambda \setminus p_8, \Lambda \setminus p_7, \Lambda \setminus p_6, \Lambda \setminus p_5, \Lambda \setminus p_4, \Lambda \setminus p_2\}$. З чого видно, що система утворюючих підструктур повинна включати продукції $p_2, p_4, p_5, p_6, p_7, p_8$. Такою системою буде утворююча система з аксіоматиками:

$$\{\Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4, \Lambda_5\}, \tag{5.20}$$

де $\Lambda_1 = \{p_1, p_3, p_8\}$, $\Lambda_2 = \{p_1, p_2, p_3, p_8\}$, $\Lambda_3 = \{p_1, p_3, p_4, p_8\}$, $\Lambda_4 = \{p_1, p_3, p_5, p_7\}$ і $\Lambda_5 = \{p_1, p_3, p_5, p_6, p_7\}$.

Множина аксіоматик задає структури виводів відповідних ланцюжків: $l_1 = abc$, $l_2 = aabc$, $l_3 = abbc$, $l_4 = abcc$, $l_5 = abccc$. Структурний граф системи утворюючих підструктур формальної структури $C(G)$ за включеннями представлено на рис. 5.2.

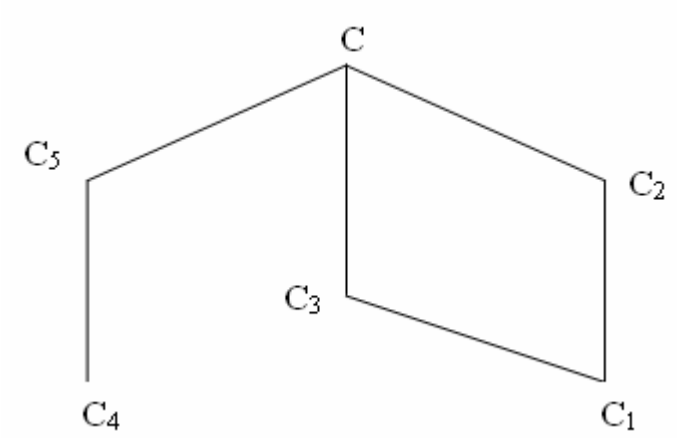


Рис. 5.2. Граф системи утворюючих підструктур структури $C(G)$

Ізольованою підструктурою відносно структури $C(G)$ серед системи підструктур з множиною аксіоматик (5.20) є структура $\overline{C}_5^*$, для якої доповненою до формальної структури $C(G)$ буде структура $\overline{C}_2^* \bigcup \overline{C}_3^*$, тому система утворюючих підструктур розбивається на класи $K_1 = \{\overline{C}_4^*, \overline{C}_5^*\}$ і $K_2 = \{\overline{C}_1^*, \overline{C}_2^*, \overline{C}_3^*\}$. Очевидно, на незалежних підструктурах цих класів можливо отримати повну систему утворюючих породжуючих підструктур $M_C^m = \{\overline{C}_2^*, \overline{C}_3^*, \overline{C}_5^*\}$. Побудована таким чином повна система утворюючих підструктур є неоднозначною тому, що за основні утворюючі підструктури можливо взяти кінцеві породжуючі структури графу C_1 і C_4 з відповідними аксіоматиками та додати до них підструктури з аксіоматиками $\Lambda_5 \setminus \Lambda_4$, $\Lambda_3 \setminus \Lambda_1$ і $\Lambda_2 \setminus \Lambda_1$. Отже отримаємо нову систему утворюючих підструктур з системою аксіоматик $\{\Lambda_1, \Lambda_4, \Lambda_2 \setminus \Lambda_1, \Lambda_3 \setminus \Lambda_1, \Lambda_5 \setminus \Lambda_4\}$. Таким чином на структурах двох простих ланцюжків l_1 і l_4 та рекурсивних продукціях p_2, p_4 і p_6 аксіоматики (5.19) відтворюється формальна граматична структура $C(G)$.

Зауваження 5.4. Запропоновану методику побудови повної системи утворюючих підструктур також зручно застосовувати у тому випадку коли відомі дерева виводів утворюючих ланцюжків, при цьому слід звернути увагу, що однозначне відтворення формальних систем можливе тільки для *VC* - структур.

5.6. Конструктивні структури контекстно-залежних граматик

Розглянемо широкий клас формальних структур обумовлених обмеженнями виразів (5.15).

5.6.1. Клас структур контекстно-залежних граматик

Як було вказано особливістю BC - структур є те, що продукції їх аксіоматик мають контекстне супроводження, яким можливо скористатися, наприклад, при розробці діалогових систем штучного інтелекту зв'язавши контекстне супроводження з певною темою «обміну даних», «розмови» тощо. Безпосередньо контекстно-залежні граматики отримали таку назву через можливість в них досліджувати граматичні структури речень природних мов. Наприклад, речення: «Операторна послідовність утворює алгоритмічну програму» може бути представлене через його мовну послідовність характеристик-складників (іменника, прийменника та дієслова)

Операторна послідовність утворює алгоритмічну програму
(прийменник) (іменник) (дієслово) (прийменник) (іменник).

Прикладами контекстно-залежної мови (BL - мови) є мова ідентифікаторів, розглянута у підпункті 5.4.1, в якій продукції аксіоматики її породжуваної структури безпосередньо задовольняють умовам (5.15), а також мова розглянута у вправі пункту 5.2 за аксіоматиками (5.8) і (5.11) та її частковий випадок $\{a^k b^k a^k;\ k \in \mathbb{N}\}$. Клас контекстно-залежних граматик взагалі то менше досліджений ніж класи безконтексних і автоматних граматик [3]. В класі BC - структур можливо виділити клас o - вільних структур, для цього дамо:

Визначення 5.18. Формальна структура $C(G)$, продукції аксіоматики якої $x \to y$ задовольняють умові $|x| \le |y|$ називається *нескороченою структурою.*

Нескладно зрозуміти, що формальна BC - структура з продукціями її аксіоматики типу (5.15) $u_i \alpha_i y_i \to u_i x_i y_i$, $\alpha_i \in N$ є нескороченою, якщо $|x_i| \ge 1$ або $x_i \ne \varepsilon$. В певному розумінні (продукції виду $u_i \alpha_i y_i \to u_i y_i$ не належить до аксіоматики) така структура породжує o - вільну BL - мову.

Теорема 5.5 У всякому класі еквівалентності KC_L з нескороченою однозначною структурою існує однозначна o- вільна BC- структура.

▷Доведемо спочатку, що в класі KC_L знаходиться o- вільна BC- структура. Для цього в класі KC_L виберемо нескорочену структуру $C(G)$, яка за теоремою 5.1 має еквівалентну нормальну структуру $C(G_h)=\langle V_h,\Sigma,\Lambda_h\rangle$, де $V_h=A\bigcup N_h$ і A – термінальний алфавіт структури $C(G)$. Нехай довільна продукція аксіоматики Λ_h має вигляд

$$p_i: x\to y;\quad x\in\mathbb{F}(N_h),\ y\in\mathbb{F}(V_h); \tag{5.21}$$

причому, очевидно, – $|x|\le|y|$, перетворимо продукцію (5.21), за результатами роботи [16] у сукупність продукцій аксіоматики Λ_0 o-вільної BC- структури. При перетворені слід врахувати можливості випадків лівих частин продукцій (5.21).

1. Розглянемо спочатку наступний виключний випадок, коли $|x|=0$, тобто $x=\varepsilon$, що відповідає продукції $\varepsilon\to y$, яка дозволяє вставляти при безпосередньому виводі з ланцюжка $z=\beta_1\beta_2\ldots\beta_n$ ланцюжок y_y перед або позаду будь якого символа β_j (нагадаємо, що для довільного ланцюжка $abc=\varepsilon a\varepsilon b\varepsilon c\varepsilon$, бо конкатенація $\varepsilon\boxtimes a\cdots=a\cdots$). Замінимо продукцію (5.21) наступною множиною:

$$\{\alpha\to\alpha y\,|\,y\alpha\},\ \text{де}\ \alpha\in N_h, \tag{5.22}$$

яка також дозволяє вставляти ланцюжок y попереду або позаду будь якого символа не термінального алфавіту N_h.

2. Якщо у виразі (5.21) $|x|=1$, тоді ця продукція зостається у аксіоматиці Λ_0 без будь яких змін.

3. У наступному випадку коли $|x|>1$, припустимо, що $x=\alpha_1\alpha_2\ldots\alpha_k,\ \alpha_j\in N_h,\ j=1,2,\ldots,k$; $y=s_1s_2\ldots s_m,\ s_j\in\mathbb{F}(V_h)$, $j=1,2,\ldots m$, тобто продукція (5.21) виглядає наступним чином $\alpha_1\alpha_2\ldots\alpha_k\to s_1s_2\ldots s_m,\ k\le m$; замінимо її рекурсивною послідовністю продукцій:

$$\left\{\begin{array}{l}
\alpha_1\alpha_2\ldots\alpha_k \to \beta_1^i\alpha_2\ldots\alpha_k,\\
\beta_1^i\alpha_2\ldots\alpha_k \to \beta_1^i\beta_2^i\alpha_3\ldots\alpha_k,\\
\qquad\vdots\\
\beta_1^i\beta_2^i\ldots\beta_{k-1}^i\alpha_k \to \beta_1^i\beta_2^i\ldots\beta_k^i,\\
\beta_1^i\beta_2^i\ldots\beta_k^i \to s_1\beta_2^i\ldots\beta_k^i,\\
s_1\beta_2^i\ldots\beta_k^i \to s_1s_2\beta_3^i\ldots\beta_k^i,\\
\qquad\vdots\\
s_1s_2\ldots s_{k-1}\beta_k^i \to s_1s_2\ldots s_{k-1}s_k\ldots s_m;
\end{array}\right\} \qquad (5.23)$$

де символи $\beta_1^i, \beta_2^i, \ldots, \beta_k^i \notin N_h$ є додатними не терміналами.

Таким чином аксіоматика Λ_0 з продукціями (5.22), (5.23) і випадку 2 відповідає нескороченій o- вільній BC- структурі, причому продукції (5.22) і (5.23) не порушують однозначності цієї структури. Покажемо тепер, що формальна структура $C(G_0)$ з аксіоматикою Λ_0 належить класові KC_L. Для цього розглянемо будь який ланцюжок u мови $L(G_h)$ нормальної структури $C(G_h)$, вивід якого є $W(u)$. Зрозуміло, що за співвідношеннями (5.22) і (5.23) виводові $W(u)$ ланцюжка завжди можливо однозначно поставити у відповідність вивід цього ж ланцюжка $W_0(u)$ у структурі $C(G_0)$, тому можливо записати, що

$$L(G_0) \subseteq L(G_h). \qquad (5.24)$$

І навпаки, всякому виводові $W_0(u)$ з рекурсивною послідовністю (5.23), до якої завжди можливо звести цей вивід пересортувавши в ньому безпосередні виводи, можливо поставити у відповідність однозначний вивід $W(u)$, який відповідає умові випадку 3 тому маємо наступне співвідношення BC

$$L(G_h) \subseteq L(G_0). \qquad (5.25)$$

Порівнюючи вирази (5.24) і (5.25) заключаємо, що структури $C(G_h)$, $C(G_0)$ еквівалентні, а так як $C(G_h) \in QC_L$ тоді також $C(G_0) \in QC_L$. $\lhd$

Клас еквівалентності BC - граматичних структур цікавий і тим, що в ньому можуть знаходитися і o - формальні граматичні структури, аксіоматики яких мають скорочені продукції $u_i\alpha_i y_i \to u_i o y_i = u_i y_i$.

Теорема 5.6. У всякому класі еквівалентності KC_L довільної структури існує o - формальна BC - структура.

▷ Ця теорема доводиться за наступною схемою: спочатку вона доводиться так же як теорема 5.5, тобто для довільної продукції (5.21) формальної структури $C(G)$ при умові $|x| \le 1$ будуються рівносильні продукції за випадками 1 і 2 теореми 5.5; у третьому випадку продукції $x = \alpha_1\alpha_2 \ldots \alpha_k \to y,\ \alpha_j \in N_h,\ j = 1,2,\ldots,k$, замість системи нескорочених продукцій (5.23) ставиться у відповідність упорядкована система продукцій

$$\left\{\begin{array}{l}
\alpha_1\alpha_2 \ldots \alpha_k \to \beta_1^i\alpha_2 \ldots \alpha_k, \\
\beta_1^i\alpha_2 \ldots \alpha_k \to \beta_1^i\beta_2^i\alpha_3 \ldots \alpha_k, \\
\qquad\qquad \vdots \\
\beta_1^i\beta_2^i \ldots \beta_{k-1}^i\alpha_k \to \beta_1^i\beta_2^i \ldots \beta_k^i, \\
\beta_1^i\beta_2^i \ldots \beta_k^i \to \beta_2^i \ldots \beta_k^i, \\
\beta_2^i\beta_3^i \ldots \beta_k^i \to \beta_3^i \ldots \beta_k^i, \\
\qquad\qquad \vdots \\
\beta_{k-1}^i\beta_k^i \to \beta_k^i, \\
\beta_k^i \to y;
\end{array}\right\}.$$

Зрозуміло, що таким чином побудована система продукцій аксіоматики відповідає o - формальній структурі. І далі доведення проводиться так же, як у теоремі 5.5. ◁

Із теореми 5.6 можливо зробити висновок про те, що мова L, відносно якої розглядався клас еквівалентності, може включати в себе порожній ланцюжок або ні. З доведень теорем 5.5 і 5.6 випливає, що до довільного класу еквівалентності KC_L належать BC - формальні структури і o - формальні BC_{BC} - структури тоді і тільки тоді, коли породжена мова не містить порожнього ланцюжка, тобто $\varepsilon \notin L$.

5.6.2. Виводи і їх структури

Перейдемо тепер до питання структур ланцюжків виведених в BC- структурах. Як було вказано у першому і другому пунктах цього розділу, структуру будь якого ланцюжка довільної однозначної мови породженої однозначною граматичною структурою можливо задати за допомогою підструктури граматичної структури або його виводу (інколи спеціальним чином розміченого) за допомогою схеми виводу, за допомогою способу виводу, при цьому ці чотири можливості відтворення структури ланцюжка еквівалентні у тому розумінні, що між ними існує взаємно однозначне (ізоморфне) відношення. Окрім того за цими способами відтворення структури ланцюжків заданої мови інколи однозначно відтворюється породжуюча граматична структура. Ці ж варіанти можливі і для BC- мов, також для BL- мов можливо визначити структуру ланцюжка за допомогою *дерева виводу* [16].

Розглянемо вивід $W(l)$ (схему виводу, спосіб виводу) довільного ланцюжка $l \in HL$, нехай його будь який безпосередній вивід

$$u_i \alpha_i v_i \mapsto u_i x_i v_i, \ \alpha_i \in N, \ u_i, v_i, x_i \in \mathbb{F}(V), \tag{5.26}$$

де ланцюжок $x_i = x_1^i x_2^i \dots x_k^i$ складається з підланцюжків x_j^i, $j \le k$, кожен з яких є не термінал або термінальний підланцюжок ланцюжка l. Поставимо тепер у відповідність безпосередньому виводові (5.26) упорядкований кортеж

$$(\alpha_i, x_1^i), (\alpha_i, x_2^i), \dots (\alpha_i, x_k^i). \tag{5.27}$$

Розташовуючи в певній послідовності за виводом $W(l)$ кортежі (5.27), отримаємо граф виводу ланцюжка l або, як його ще називають C- *маркер* [16]. Вивід $W(l)$ правильний, він завжди починається з аксіоми пріоритету аксіоматики BC- структури і граф виводу утворює *дерево виводу*, коренем якого є не термінал лівої частини аксіоми пріоритету з якого починається вивід ланцюжка $W(l)$, його вершинами будуть не термінали або термінальні підланцюжки ланцюжка l.

Для прикладу розглянемо структуру з наступною аксіоматикою:

$$\begin{cases} \sigma \to \alpha\alpha\beta - \text{аксіома приорітету}, \\ \alpha\beta \to \delta\beta\beta, \\ a\beta\beta \to ab\beta, \\ \beta \to \gamma, \\ \text{аксіоми виводу}: \\ \alpha \to a, \\ \delta \to a, \\ \gamma \to c; \end{cases}$$

і наведемо вивід ланцюжка $aabc$ в цій структурі

$$W(aabc) = (\sigma, \alpha\alpha\beta, a\alpha\beta, a\delta\beta\beta, aa\beta\beta, aab\beta, aab\gamma, aabc).$$

Дерево виводу (рис. 5.3) для цього випадку буде мати корінь з позначкою σ і $st(\sigma) = 3$, внутрішні вершини з позначками: $\alpha, \beta, \delta, \gamma$ мають відповідні степені $st(\alpha) = 2 \mid 3$, $st(\beta) = 2$, $st(\delta) = 2$, $st(\gamma) = 2$ і чотири кінцеві вершини з термінальними позначками: a, a, b, c.

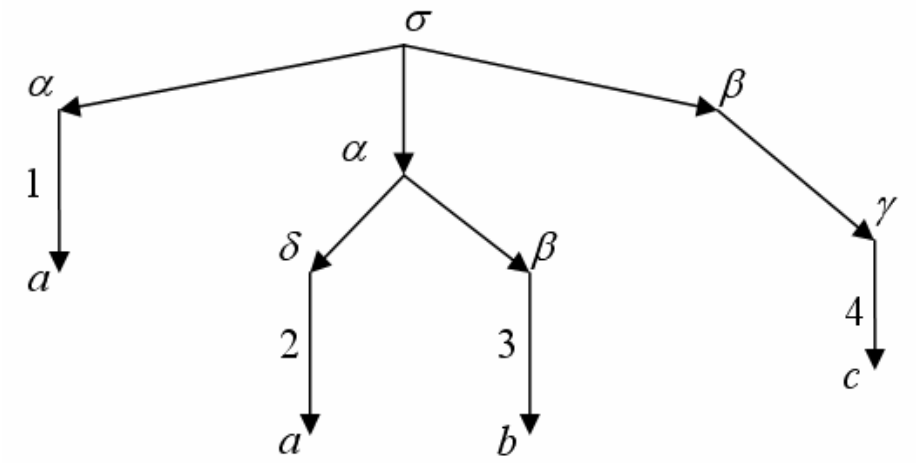


Рис. 5.3. Дерево виводу ланцюжка $aabc$

Вище встановлене відношення між виводом ланцюжка l і його деревом виводу є взаємно однозначним, хоча відношення між безпосереднім виводом (5.26) і кортежем (5.27) визначене з точністю до контекстного супроводження безпосереднього виводу (5.26), однак вивід $W(l)$, побудований на безпосередніх виводах (5.26), враховує контекстне супроводження (якщо його коректно виділити у ланцюжках виводу) при послідовному виведені ланцюжків виводу $W(l)$. Таким чином за властивістю транзитивності відношень між схемою

виводу (способом виводу) та розміченим деревом виводу існує взаємно однозначне відношення, цей результат дозволяє стверджувати, що

Твердження 5.9. Визначення структури ланцюжка BL- мови за його виводом, схемою виводу, способом виводу і деревом виводу еквівалентні в тому розумінні, що структура ланцюжка частково відтворює BC- формальну граматичну структуру.

Задати структуру ланцюжка BL- мови також можливо за допомогою входжень його підланцюжків [16]. Нехай ланцюжок x є підланцюжком деякого ланцюжка $l = uxv$, а $r = |u| + 1$ місце, з якого починається розташовувати ланцюжок x в ланцюжку l.

Визначення 5.19. Упорядкована пара (x, r) називається *входженням підланцюжка x в ланцюжок l* починаючи з символа r ланцюжка l (при рахуванні з ліва на право).

Наприклад, для ланцюжка $l = abcdab$ його підланцюжки $l_1 = a$, $l_2 = ab$ і $l_3 = cdab$ утворюють входження: $(l_1,1)$, $(l_1,5)$, $(l_2,1)$, $(l_2,5)$ і $(l_3,3)$ в ланцюжок l.

Між різними входженнями підланцюжків в один і той же ланцюжок існують певні відношення. Так, якщо підланцюжкам x і y ланцюжка l відповідають входження (x,r) і (y,s) причому $r \le s \le r + |x| - 1$ або $s \le r \le s + |y| - 1$, тоді входження *перекриваються*. Якщо ж ланцюжок y є підланцюжком ланцюжка x і $r \le s \le s + |y| - 1 \le r + |x| - 1$, то входження (x,r) *покриває* входження (y,s). Для прикладу, входження $(x,1)$ і $(y,2)$ підланцюжків $x = ab$ і $y = bcd$ ланцюжка $l = abcdab$ перекриваються тому, що при значеннях $r = 1$ і $s = 2$ виконується умова $1 \le r \le 1 + 2 - 1$, а входження $(x,1)$ та $(y,5)$ не перекриваються; входження ж $(abcd,1)$ покриває входження $(bcd,2)$ цього ланцюжка l виходячи з того, що виконується умова $1 \le s \le 2 + 3 - 1 \le 1 + 4 - 1$.

Розглянемо множину $D(l)$ всіх різноманітних входжень ланцюжка l деякої мови $L(G)$, породженої граматичною структурою $C(G)$. До речі порожнє входження $(\varepsilon,\cdot)$ не включається до множини $D(l)$ тому, що місце цього входження не визначене. Очевидно, у множині входжень $D(l)$ завжди можливо виділити певним чином упорядковану підмножину, яка задає будову ланцюжка l. Наприклад, можливо виділити підмножину входжень у $D(l)$ з фіксованим першим або іншим

місцем у входженнях таких, що всі вони задовольняють умові покриття. Так для ланцюжка $l = abcdab$ його будова може бути визначена підмножиною входжень

$$\{(a,1),\ (ab,1),\ (abc,1),\ (abcd,1),\ (abcda,1),\ (abcdab,1)\}.$$

Також поширена інша методика виділення підмножини складників [16, 4] з множини входжень $D(l)$.

Визначення 5.20. Множина $C(l) \subset D(l)$ називається *системою складників* ланцюжка $l = a_1 a_2 \cdots a_n$, якщо її входження задовольняють наступним умовам:

1) власне входження $(l,1) \in C(l)$;

2) всі одноелементні входження $(a_i, i) \in C(l),\ i = 1,2,\ldots,n$;

3) для будь яких двох входжень $(x,r),\ (y,s) \in C(l)$ вони або покриваються, або не перекиваються.

За визначенням 5.17 система складників ланцюжка є не однозначною. Дійсно будову ланцюжка $l = ababc$ можливо задати так

$$C_1(l) = \{(a,1),\ (b,2),\ (a,3),\ (b,4),\ (c,5),\ (ab,1),\ (ab,3),\ (ababc,1)\}$$

або наступним чином

$$C_2(l) = \{(a,1),\ (b,2),\ (a,3),\ (b,4),\ (c,5),\ (ab,1),\ (bc,4),\ (ababc,1)\}.$$

Не однозначність визначення синтаксичної будови ланцюжків за допомогою системи складників частково вирішується для заданої граматичної структури, в якій ці ланцюжки породжуються. Нехай розглядається граматична BC- структура і породжена нею мова $L(G)$, для якої ланцюжок $l \in L(G)$ має дерево виводу (C- маркер). Тоді на дерево виводу будемо дивиться як на множину підграфів виводу підланцюжків ланцюжка l, кожен з яких розгортає не термінальні символи до кінцевих термінальних символів складників підланцюжків ланцюжка l. Сукупність таким чином розгорнутих ланцюжків і утворюють множину ланцюжків, яка відтворюють систему складників. Якщо ввести певний порядок у ланцюги виводів (розгортання) вздовж гілок дерева пронумерувавши їх певним чином, наприклад, так як це зроблено для дерева рис. 5.3, тоді отримаємо для ланцюжка $l = aabc$ систему складників

$$S(l) = \{(a,1),\ (a,2),\ (b,3),\ (c,4),\ (ab,2),\ (a^2bc,1)\}.$$

Крім вище наведеного прийому виділення системи складників існують і інші методи їх побудови. Наприклад, у лінгвістиці [4] застосовується наступний метод розмітки системи складників. Кожному символу виведеного в граматичній структурі ланцюжка ставиться у відповідність позначка (атрибут). В якості атрибуту вибирається відповідна не термінальна позначка вершини дерева виводу, із якої виводиться цей символ. При цьому атрибут записується окремо за дужками, якими виділяється входження. Отже, для наведеного на рис. 5.3 дерева виводу така система складників має вигляд

$$\tilde{S}(l) = ((a)\alpha\ ((a)\delta\ (b)\beta)\alpha\ ((c)\gamma)\beta)\sigma,$$

де не термінальні символи $\alpha, \beta, \delta, \gamma, \sigma$ – позначки, які відповідають входженням $(a)\alpha$, $(a)\delta$, $(b)\beta$, $((a)\delta(b)\beta)\alpha$ і так далі.

Розглянемо приклад побудови системи складників для визначення структури речення: $l =$ «еліпс перетинає параболу».

Нехай словник підструктури заданого речення граматичної структури мови складається з символів $\{e, p, r, \sigma, \alpha_{\div,\hat{i},\acute{i}}, \alpha_{\acute{c},\hat{i},\c{c}}, \beta^3, \gamma^3\}$, де через символ e позначено слово «еліпс», через – p позначено «перетинає», слово «парабола» позначене символом r; сигнатура – $\{\rightarrow^2, =^2\}$ і аксіоматика структури виводу цього речення наступна

$$\Lambda = \begin{cases} e = \textit{еліпс}, p = \textit{перетинає}, r = \textit{параболу}; \\ \sigma = \textit{речення}; \\ \alpha_{\text{ч, о, н}} = \textit{іменник} - \textit{чоловічий}, \textit{однина}, \textit{називний}; \\ \alpha_{\text{ж, о, з}} = \textit{іменник} - \textit{жіночий}, \textit{однина}, \textit{знахідний}; \\ \beta^3 = \textit{група дієслова у третій особі}; \\ \gamma^3 = \textit{перехідне дієслово у третій особі}; \\ \sigma \rightarrow \alpha_{\text{ч, о, н}}\ \beta^3 - \textit{аксіома приорітету}; \\ \beta^3 \rightarrow \gamma^3 \alpha_{\text{ж, о, з}}; \\ \textit{аксіоми виводу}: \\ \alpha_{\text{ч, о, н}} \rightarrow e, \\ \alpha_{\text{ж, о, з}} \rightarrow r, \\ \gamma^3 \rightarrow p; \end{cases} \qquad (5.28)$$

тоді система складників ланцюжка l запишеться так:

$$((e)\alpha_{\div,\hat{\imath},\acute{\imath}}((p)\gamma^3\ (r)\alpha_{\acute{c},\hat{\imath},\varsigma})\beta^3)\sigma$$

або

$$((\textit{еліпс})\alpha_{\textit{ч, о, н}}((\textit{перетинає})\gamma^3\ (\textit{параболу})\alpha_{\textit{ж, о, з}})\beta^3)\,\textit{речення}.$$

Від системами складників $C(l)$ завжди можливо перейти до системи $\tilde{C}(l)$ і навпаки – систему складників $\tilde{C}(l)$ можливо записати як $C(l)$.

Виходячи з того, що між цими системами складників існує взаємнооднозначне відношення, будемо вважати їх еквівалентними $C(l) \sim \tilde{C}(l)$. Частково відтворена за ланцюжком l BC- структура є BC_l- підструктурою структури $BC(G)$. Отже за системами складових та відповідним їм деревами виводів (при заданому способі їх обходу) ланцюжків BL- мови можна однозначно відтворити BC- формальну граматичну структуру, тобто $\bigcup_l BC_l = BC(G)$. В свою чергу об'єднання $\bigcup_l BC_l$ відтворює структуру формальної BC- структури, що можливо пов'язати з системою утворюючих ланцюжків l або повною системою BC_l- підструктур, за допомогою яких повністю відтворюється $BC(G)$- структури.

Твердження 5.10.. Визначення структури ланцюжка BL- мови за його виводом, схемою виводу, способом виводу, деревом виводу і системою складників еквівалентні в тому розумінні, що структура ланцюжка частково відтворює BC- граматичну структуру.

5.6.3. Завдання і вправи

1. . Довести, що формальні мови $L(G)$ є BL- мови, якщо:

   а) $\{a^n b^m a^n;\ n \in \mathbb{N},\ m \in \mathbb{N}_0\}$,

   б) $\{1^n 0^m 1^m 0^n;\ n \in \mathbb{N},\ m \in \mathbb{N}_0\}$,

   в) $\left\{\sqrt{a^n + b^n + a^n};\ n \in \mathbb{N}\right\}$,

   г) $\left\{\ln\sqrt{(x^m * y^m)^k};\ k, m \in \mathbb{N}\right\}$,

   д) $\{a^n b^n a^m c^m;\ n \in \mathbb{N},\ m \in \mathbb{N}_0\}$,

е) $\{\sin^n(x^m + y^m)^n;\ n,m \in \mathbb{N}\}$,

ж) $\{a^n b^k c^n d^k;\ k,n \in \mathbb{N}\}$.

2. Побудувати формальні структури типу 1 за класифікацією Хомського на термінальному алфавіті $A = \{a,b,c\}$ якщо

а) $\{a^n b^k c^{2n};\ k,n \in \mathbb{N}_0,\ k+n>0\}$,

б) $\{a^{2k} c^{2k} b^n;\ k \in \mathbb{N},\ n \in \mathbb{N}_0\}$,

в) $\{a^k b^n c^{2n} a^k;\ k,n \in \mathbb{N}_0,\ k+n>0\}$,

г) $\{a^{2k} b^n c^{m+1} b^n;\ k,n,m \in \mathbb{N}_0\}$,

д) $\{a^n c^{n+1} b^n;\ n \in \mathbb{N}_0\}$.

3. Записати аксіоматику формальної структури у вигляді нормальної форми Бекуса – Наура для наступних мов:

а) поштовий індекс,
б) поштова адреса,
в) дійсне число з фіксованою крапкою для мови C,
г) дійсне число з плаваючою крапкою для мови C,
д) дійсне число без знаку, яке припускається в мові C.

4. Чи можливо побудувати BC- структуру еквівалентну до формальної структури граматики з системою продукцій (σ – початок):

$$\{\sigma \to aab\gamma a,\ \gamma \to ba,\ \sigma \to a\sigma\alpha,\ a\alpha \to \alpha a,\ \gamma\alpha \to b\gamma a\}?$$

Якщо можливо, то побудувати таку структуру.

5. Для правильних ланцюжків l, мов задачі 1, довжина яких $|l| \geq 7$, побудувати схеми виводів, спосіб виводів, дерева виводів, системи складників.

6. Виконати завдання вправи 5 для мов завдання 2.

7. Для ланцюжків: $(x+y)*\sin x$, $2y*e^x$, $\sqrt{x-y}*\ln(xy)$, $(tgx-\cos y):(z+1)$ побудованих на термінальному алфавітові $A = \{x,y,z,\sin,\sqrt{},\ln,tg,\cos,e,1,2,+,-,*,:,(,)\}$, вказати пари входжень, котрі перекриваються, покриваються, не перекриваються.

8. Для ланцюжків вправи 7 побудувати системи складників $C(l)$.

9. Для відомих «ключових слів» мови програмування C, як ідентифікаторів побудувати системи складників.

10. Нехай ланцюжок x побудований на термінальному алфавіті $A=\{a_i\}_{i=1}^{n}$. Для мови $L(G)$ відтворити BC - структуру, і для довільних ланцюжків $l_j \in L(G)$ їх дерева виводів та системи $C(l_j)$, якщо $L(G)=\{xx';\ x\in \mathbb{F}(A),\ x\neq\varepsilon,\ xx'=x\otimes x'\}$; де $\otimes$ – операція конкатенації, x' – ланцюжок, транспонований до ланцюжка x.

11. Нехай $C(G)=\langle\{\wedge,\vee,\neg,a,(,)\}\cup\{\sigma,\beta,\delta,\nu,\eta\},\Sigma,\Lambda\rangle$ формальна структура логіки висловлювань відносно a, для структури якої аксіоматика

$$\Lambda=\begin{cases}\sigma\to\beta,\ \sigma\to\delta-\textit{аксіоми приорітету};\\ \beta\to\delta,\ \ \beta\to\delta\wedge\beta,\ \ \delta\to\beta,\ \ \delta\to\beta\vee\delta;\\ \beta\to\nu,\ \ \beta\to\nu\wedge\beta,\ \ \delta\to\nu,\ \ \delta\to\nu\vee\delta;\\ \nu\to\eta,\ \ \nu\to\neg\eta;\\ \eta\to(\sigma);\\ \eta\to a-\textit{аксіома виводу}.\end{cases}$$

Яка мова логіки висловлювань породжується цією структурою і якого типу?

Довести, що структура неоднозначна. Виділити однозначну підструктуру у структурі $C(G)$, котра породжує ту ж саму мову.

12. Дослідити структуру $C(G)$ на однозначність, якщо

$$G=\left\langle\begin{array}{l}\{\underline{якщо},\ \underline{тоді},\ \underline{інакше},Б\},\{\sigma,\varsigma,\xi,\alpha\},\sigma,\\ \{\sigma\to\varsigma,\ \sigma\to\varsigma\ \underline{інакше}\ \alpha,\ \varsigma\to\xi\alpha,\\ \xi\to\underline{якщо}\ Б\ \underline{тоді},\ \alpha\to\sigma,\ \alpha\to Б\}\end{array}\right\rangle$$

Мова якого типу породжується за цією структурою? Побудувати дерево виводу для вибраного ланцюжка цієї мови.

13. Для ланцюжків a^2ba, a^2ba^2, a^3ba^2 виведених у структурі $C(G)$ граматики $G=\langle\{a,b\},\{\alpha,\sigma\},\sigma,\{\sigma\to a\alpha a,\alpha\to\alpha a,\alpha\to a\alpha,\alpha\to b\}\rangle$ визначити:

а) вивід ланцюжка,
б) довжину виводу,
в) схему виводу,
г) спосіб виводу,
д) дерево виводу,

е) підструктуру, в якій виводиться ланцюжок a^2ba;

ж) системи складників $C(l)$ і $\tilde{C}(l)$.

14. За наведеним виводом ланцюжка $l = abcbacabbaa$

$W(l) = (\sigma, ab\alpha b\beta, ab\beta a\alpha b\beta, ab\sigma ba\alpha b\beta, abcba\alpha b\beta, abcbacabb\beta, l)$:

а) відтворити повну формальну структуру,

б) визначити тип відтвореної структури,

в) записати спосіб проведення виводу,

г) побудувати системи складових $C(l)$ і $\tilde{C}(l)$.

15. Спираючись на аксіоматику (5.27) побудувати розмічену систему складових $\tilde{C}(l)$ для наступного тексту «Гіперповерхня проектується на простір. Проекція відображається у пряму. Пряма трансформується у точку».

16. Для повних структур з аксіоматиками

$$\Lambda_1 = \begin{cases} \alpha \to x, \\ \alpha \to \alpha x \end{cases}; \quad \Lambda_2 = \begin{cases} \alpha \to x, \\ \alpha \to x\alpha \end{cases} \text{ і } \quad \Lambda_3 = \begin{cases} \alpha \to x, \\ \alpha \to x\alpha x \end{cases}$$

а) визначити тип еквівалентності структур,

б) визначити способи виводу,

в) побудувати дерева виводів,

г) побудувати системи звичайних і розмічених складників.

5.7. Контекстно-вільні граматики і їх структури

Контекстно-вільні граматики завдяки їх застосуванню у різних аспектах програмної інженерії, зокрема мовах програмування, теоріях синтаксичного розбору, побудови трансляторів та ін. набули більшого поширення і розвитку. Цим граматикам присвячено великий спектр літературних джерел (див., наприклад, [1 – 4, 9 – 12, 14, 16]).

5.7.1. Класи і моделі на *VC*- структурах

Нагадаємо, що за класифікацією Хомського контекстно-вільні граматики і відповідні ним *VC*- структури мають продукції типу $\alpha \to x$; $\alpha \in N$, $x \in \mathbb{F}(V)$. Таким чином *VC*- структура є частковим ви-

падком BC- структури, продукції аксіоматики якої мають порожнє контекстне супроводження. Контекстно-вільні структури породжують контекстно-вільні VL- мови. Так формальна мова $\{a^n b^k c^n;\ n \in \mathbb{N},\ k \in \mathbb{N}_0\}$ є VL- мовою, дійсно вона породжується VC- структурою з аксіоматикою:

$$\Lambda = \begin{cases} \textit{аксіоми початку}: \\ \sigma \to a\alpha c, \\ \sigma \to ac - \textit{аксіома виводу}; \\ \alpha \to a\alpha c, \\ \alpha \to b\beta, \\ \beta \to b\beta, \\ \textit{аксіоми виводу}: \\ \alpha \to ac, \\ \alpha \to b, \\ \beta \to b. \end{cases} \tag{5.29}$$

Класи контекстно-вільних граматик і їх структур досить гарно прилаштовані до опису конструкцій штучних мов програмування. От як це зроблено для мови типу АЛГОЛ американським математиком Бекусом за допомогою так званих *нормальних форм Бекуса – Наура*. Нормальна форма задається словником, який складається з термінальних символів: букв, цифр, знаків операцій, роздільників, дужок, ключових слів (*if*, *for*, *real*, *procedure* тощо) та не терміналів (метазмінних), один з яких визначений як початковий символ і множиною продукцій у вигляді металінгвистичних формул. Металінгвістичні формули будуються на словникові за допомогою металінгвістичних зв'язок: (::=) – «рівне за визначенням» , (|) – «або», причому метазмінні у формулах позначають кутовими дужками ($\langle,\rangle$). Наведемо деякі приклади таких формул, з майже 150 металінгвістичних формул граматики мови типу АЛГОЛ:

1. $\langle \textit{буква} \rangle ::= a \mid b \mid c \mid ... \mid z \mid ...$, де символ (…) застосовано для скорочення запису.
2. $\langle \textit{цифра} \rangle ::= 0 \mid 1 \mid ... \mid 9$.
3. $\langle \textit{ціле без знаку} \rangle ::= \langle \textit{цифра} \rangle \mid \langle \textit{ціле без знаку} \rangle \ \langle \textit{цифра} \rangle$.

4. ⟨*ідентіфікатор*⟩ ::= ⟨*буква*⟩ | ⟨*ідентіфікатор*⟩⟨*буква*⟩ | ⟨*ідентіфікатор*⟩⟨*цифра*⟩.
5. ⟨*змінна*⟩ ::= ⟨*ідентифікатор*⟩ і так далі.

До речі, в багатьох граматиках мов програмування системи продукцій представляються у нормальній формі Бекуса – Наура.

5.7.2. Граматична структура мов програмування на основі систем рівнянь

В програмуванні формульне моделювання мовних конструкцій дозволяє виконувати алгебраїчні перетворення програм [16]. Розглянемо один із підходів заснований на системі рівнянь. На представлення продукцій у нормальній формі Бекуса – Наура практично одночасно звернули увагу дослідники Л. Ханес з Массачусетського технологічного інституту та В. Редько з Київського інституту кібернетики і запропонували абстрактну математичну модель мови типу АЛГОЛ у вигляді системи рівнянь.

Наведені у пункті 5.7.1 металінгвістичні формули визначають множини: букв – B, цифр – C, ідентифікаторів – I, цілих без знаку – D, змінних – Z, замінивши символи зв'язок (::=), (|) на звичайні математичні символи відповідно (=) і $\cup$, отримаємо математичну модель метазмінних у вигляді формул – рівнянь:

$$D = C \cup DC = f_1(C, D),\ I = B \cup IB \cup IC = f_2(B, I, C),\ Z = I = f_3(I). \tag{5.30}$$

Операції множення і об'єднання над множинами, які застосовуються у формулах мають відомі властивості, тому за цими властивостями можуть бути виконані перетворення формул, наприклад, формулу 1) можливо переписати як $D = DC \cup C$.

Розглянемо узагальнений процес побудови граматичної структури C_V на системі рівнянь типу (5.30).

Нехай задана повна VC- граматична структура $C(G) = \langle V, \Sigma, \Lambda \rangle$ з не термінальним алфавітом $N = \{\alpha_1, \alpha_2, ..., \alpha_k\}$ і $\Sigma = \{\rightarrow^2\}$, для аксіоматики якої продукції мають вигляд

$$p_i : \alpha_i \rightarrow x_{ij} \in \Lambda;\ j = 1, 2, \ldots, m_i. \tag{5.31}$$

Розширимо сигнатуру Σ додавши до неї дві бінарні операції: $(:=^2)$ – «присвоєння» і $(\bigcup^2)$ – «об'єднання» і виключимо з неї операцію $\{\rightarrow^2\}$, отже отримаємо $\Sigma_1 = \{:=^2, \bigcup^2\}$.

У аксіоматику Λ_V включимо формули

$$\alpha_i := f_i(\alpha_1, \alpha_2, \ldots, \alpha_k),\ i = 1, 2, \ldots, k, \tag{5.32}$$

які об'єднують продукції (5.31) по індексу j, тобто $f_i(\alpha_1, \alpha_2, \ldots, \alpha_k) = \bigcup_{j \le m_i} x_{ij}$, серед формул (5.32) за аксіоми початку приймемо ті, що відповідають продукціям аксіом початку аксіоматики Λ і також включимо до аксіоматики Λ_V властивості комутативності та асоціативності операції об'єднання. Слід звернути увагу на те, що побудована аксіоматика на формулах (5.32) не обчислює (породжує) взагалі формальну мову з двох причин: формули (5.32) побудовані на вільних змінних α_i і аксіоматика не містить правил «обчислень» за цими формулами (операція заміщення тут відсутня). Тому у сигнатуру включимо ще одну k - місцину операцію рекурсивної суперпозиції S_f^k, таким чином, що $\Sigma_V = \Sigma_1 \bigcup \{S_f^k\}$ і аксіоматику Λ_V доповнимо правилами відносно цієї операції

$$\begin{aligned} &\alpha_i^{n+1} = S_{f_i}^k\left(f_1, f_2, \ldots f_k\right) = \\ &= f_i\left(f_1(\alpha_1^n, \alpha_2^n, \ldots, \alpha_k^n), f_2(\alpha_1^n, \alpha_2^n, \ldots, \alpha_k^n), \ldots, f_k(\alpha_1^n, \alpha_2^n, \ldots, \alpha_k^n)\right), \end{aligned} \tag{5.33}$$

за якими розв'язання системи рівнянь починається з присвоєння вільним змінним порожнього значення, тобто $\alpha_i^0 = o;\ o \in A,\ i = 1, 2, \ldots, k$ (нагадаємо, що порожній символ $o \in A$ – термінальний алфавіт).

Отже нам вдалося на заданій формальній граматичній структурі побудувати структуру з системою рівнянь (5.32), очевидно, і навпаки за структурою C_V, відтворити граматичну структуру $C(G)$, тобто між залежностями (5.30) і (5.31) існує взаємно однозначне відношення відтворення структуру $C(G)$.

Наприклад, аксіоматика (5.29) граматичної структури $C(G)$ $C(G)$ перетворюється у наступну систему рівнянь (5.32):

$$\begin{cases} \sigma := f_1(a,c,\alpha) = a\alpha c \cup ac - \textit{аксіоми початку}, \\ \alpha := f_2(a,b,c,\alpha,\beta) = a\alpha c \cup b\beta \cup ac \cup b, \\ \beta := f_3(b,\beta) = b\beta \cup b. \end{cases} . \qquad (5.34)$$

Розглянемо питання еквівалентності структур $C(G)$ і C_V, тобто чи породжують вони одну і ту ж формальну мову. Для цього спочатку необхідно з'ясувати, як будується мова у формальній структурі C_V. Розглянемо розгорнуту схему алгоритму обчислень за правилом операції суперпозиції (5.33):

1) задамо вектор порожніх значень розміру k, $(o,o,\ldots,o)$ та вільним змінним α_i надамо значення цього порожнього вектора $a_i^0 = o$, в результаті отримаємо вектор $\alpha^0 = (\alpha_1^0, \alpha_2^0, \ldots, \alpha_k^0)$;

2) надамо вільним змінним α_i значення α_i^0 і знайдемо значення функцій f_i при $\alpha_i = \alpha_i^0$ так, що $f_i(\alpha_1^0, \alpha_2^0, \ldots, \alpha_k^0) = \alpha_i^1$, отже отримаємо новий вектор значень вільних змінних $\alpha^1 = (\alpha_1^1, \alpha_2^1, \ldots, \alpha_k^1)$;

3) повторюючи виконання дій за пунктом 2) для вектора значень α^1 отримаємо новий вектор значень $\alpha^2 = (\alpha_1^2, \alpha_2^2, \ldots, \alpha_k^2)$ і далі продовжуючи цей процес отримаємо на j - му кроці $\alpha^j = (\alpha_1^j, \alpha_2^j, \ldots, \alpha_k^j)$.

Визначення 5.21. Вектор $\alpha^\infty = (\alpha_1^\infty, \alpha_2^\infty, \ldots, \alpha_k^\infty)$, де $\alpha_i^\infty = \bigcup_{j=1}^{\infty} \alpha_i^j;\ i = 1,2,\ldots,k$ називається *розв'язком системи рівнянь* (5.32).

Розв'язок системи рівнянь структури C_V утворює мову породжену цією структурою.

Для системи рівнянь (5.34) структури C_V за вище наведеною схемою послідовно отримаємо вектори:

– $(\alpha^0, \beta^0, \sigma^0)$, де
  $\alpha^0 := f_2(a,b,c,o,o) = \{ac\} \cup \{b\}$,
  $\beta^0 := f_3(b,o) = b$,
  $\sigma^0 := f_1(a,c,o) = ac$;

– $(\alpha^1, \beta^1, \sigma^1)$, де
  $\alpha^1 := f_2(a,b,c,\alpha^0,\beta^0) = \{b, bb, ac, abc, a^2c^2\}$,

$\beta^1 := f_3(b, \beta^0) = \{b, bb\}$,

$\sigma^1 := f_1(a, c, \alpha^0) = \{ac, a^2c^2, abc\}$;

– $(\alpha^2, \beta^2, \sigma^2)$, де

$\alpha^2 := f_2(a, b, c, \alpha^1, \beta^1) = \{b^k, a^k c^k, a^n bc^n, ab^n c;\ k = 1,2,3,\ n = 1,2\}$,

$\beta^2 := f_3(b, \beta^1) = \{b, b^2, b^3\}$,

$\sigma^2 := f_1(a, c, \alpha^1) = \{a^n b^k c^n;\ n = 1,2,3,\ k = 0,1,2\}$;

…

– $(\alpha^\infty, \beta^\infty, \sigma^\infty)$, де $\sigma^\infty := \{a^n b^k c^n;\ n \in \mathbb{N},\ k \in \mathbb{N}_0\}$.

Отже структура C_V з системою рівнянь (5.34) відтворює формальну мову $\{a^n b^k c^n;\ n \in \mathbb{N},\ k \in \mathbb{N}_0\}$.

5.8. Лінійні граматичні структури

Найпростішими за правилами підстановок є лінійні граматики, які породжують найбідніший за потужністю клас відповідних мов. Лінійні граматики є частковим випадком контекстно-вільних граматик.

5.8.1. Типи лінійних граматик

Продовжимо розгляд обмежень, які накладаються на систему продукцій аксіоматики граматичної структури VC.

Продукцію p_i аксіоматики Λ структури $C(G)$ назвемо *лінійною*, якщо $p_i : \alpha \to x$ або $p_i : \alpha \to u\beta y$, де $\alpha, \beta \in N$; $x, u, y \in \mathbb{F}(A)$, причому може бути $u = \varepsilon$ або $y = \varepsilon$. VC – структура називається *лінійною структурою*, якщо кожна продукція її аксіоматики лінійна. Клас лінійних граматичних структур позначимо $VC_л$. Звичайно, що $VC_л \subset VC$ Відповідна мова, яка породжується лінійною структурою називається лінійною. Наприклад, граматична структура з аксіоматикою

$$\Lambda = \begin{cases} p_1 : \alpha \to a\beta b, – \textit{аксіома початку}; \\ p_2 : \beta \to a\beta b; \\ p_3 : \beta \to ab; \end{cases}$$

є лінійною але структура, для якої

$$\Lambda_1 = \begin{cases} \textit{аксіоми початку}: \\ p_1 : \alpha \to \beta\alpha\gamma, \\ p_2 : \alpha \to ab; \\ p_3 : \beta \to a; \\ p_4 : \gamma \to b; \end{cases}$$

нелінійна, тому що продукція p_1 не є лінійною. Хоча ці дві структури граматик породжують одну і ту ж мову $L(A) = \{a^n b^n;\ n \geq 2,3,\ldots\}$.

В класі лінійних граматичних структур $KC_{ë}$ можна виділити підкласи праволінійних $VC_{лп}$ і ліволінійних структур $VC_{лл}$, таких що їх продукції задовольняють умовам:

$$p_i = \begin{cases} \alpha \to x\beta; \\ \alpha \to y;\ \alpha, \beta \in N,\ x, y \in \mathbb{F}(A); \end{cases} \text{– для праволінійних структур,}$$

$$p_i = \begin{cases} \alpha \to \beta x; \\ \alpha \to y;\ \alpha, \beta \in N,\ x, y \in \mathbb{F}(A); \end{cases} \text{– для ліволінійних структур.} \qquad (5.35)$$

Прикладами праволінійних і ліволінійних структур є структури граматик з відповідними аксіоматиками

$$\Lambda_n = \begin{cases} \textit{аксіоми початку}: \\ \sigma \to a\sigma, \\ \sigma \to b; \end{cases} \qquad \text{і } \Lambda_л = \begin{cases} \textit{аксіоми початку}: \\ \sigma \to \alpha b, \\ \sigma \to b; \\ \alpha \to \alpha a; \\ \alpha \to a. \end{cases}$$

Нескладно бачити, що ці структури граматик породжують однакову мову $L(A) = \{a^k b;\ k \in \mathbb{N}_0\}$, тобто вони еквівалентні. Виникає питання відносно еквівалентності будь яких праволінійних і ліволінійних структур.

Лема 5.3. Мова $\{xy^n;\ n \in \mathbb{N},\ x, y \in \mathbb{F}(A)\}$ – праволінійна.

▷ Структура граматики, яка породжує цю мову повинна бути задана аксіоматикою з праволінійними продукціями, наприклад, такою:

$$\Lambda_n = \begin{cases} \sigma \to x\alpha, \text{ аксіома початку}; \\ \alpha \to y\alpha; \\ \alpha \to y. \end{cases} \quad \triangleleft$$

Наслідок 5.3. Будь яка мова типу

$$\{u_1 v_1^{k_1} u_2 v_2^{k_2} \cdots u_n v_n^{k_n};\ k_i \in \mathbb{N},\ u_i, v_i \in \mathbb{F}(A),\ i = \overline{1,n}\} \tag{5.36}$$

– праволінійна тому, що її можна розбити на блоки $(u_i v_i^{k_i})$, котрі породжуються структурою граматики за лемою 5.3 і зв'язок суміжних блоків виконати за допомогою продукції $\alpha_i \to v_i \alpha_{i+1}$. Тобто фрагментом аксіоматики праволінійної структури буде

$$\begin{cases} \alpha_{i-1} \to u_i \alpha_i; \\ \alpha_i \to v_i \alpha_i; \quad . \\ \alpha_i \to v_i \alpha_{i+1}; \end{cases}$$

Таким чином, надавши індексові i пробігати значення від 1 до n та прийнявши продукцію $\alpha_0 \to u_1 \alpha_1$ за аксіому початку і продукцію $\alpha_{n+1} \to v_n$ за аксіому тупікового виводу отримаємо аксіоматику структури, яка породжує мову (5.36).

Теорема 5.7. Кожна праволінійна мова типу (5.36) також є ліволінійною.

▷ Виходячи з наслідку 5.2 для мови (5.36) достатньо виконати перетворення продукцій аксіоматики леми 5.3 до еквівалентної ліволінійної структури

$$\Lambda_л = \begin{cases} \sigma \to \alpha y, \text{ аксіома початку}; \\ \alpha \to \alpha y; \\ \alpha \to x. \end{cases} \quad \triangleleft$$

Так як за теоремою 5.7 відповідні структури з класів $VC_{лп}$ і $VC_{лл}$ еквівалентні, тому будемо розглядати в подальшому клас праволінійних структур. У класі $VC_{лп}$ виділимо підклас граматичних структур, для яких на продукції (5.35) накладено умови: $|x|=1$ і $|y|=1$. Граматичні структури цього підкласу називаються автоматними і позначаються AC.

Розглянуті лінійні граматичні структури утворюють ланцюг за включенням $AC \subset VC_{лп} \subset VC_{л} \subset VC$.

5.8.2. Автоматні граматичні структури

Автоматні граматики відіграють важливу роль у формалізації алгоритмів, формальному представлені структур даних, граматиках мов програмування, тощо.

Наприклад, розглянемо формалізацію типу даних «ціле без знаку». Нехай метамовний вираз «ціле без знаку» є не термінал σ і вираз «число» є – α, тоді тип «ціле без знаку» визначається структурою автоматної граматики із аксіоматикою:

$$\Lambda = \begin{cases} \textit{аксіоми початку}: \\ \sigma \to \alpha, \\ \sigma \to \alpha\sigma; \\ \alpha = 0\,|\,1\,|\cdots|\,9, \ \textit{аксіома виводу}. \end{cases}$$

Структура граматики «ціле без знаку» породжує будь які числові ланцюжки довжини 1, 2,…

Автоматну граматичну структуру можна пов’язати з не детермінованим автоматом (див. п. 4.2.2). В цьому випадку будемо казати, що автомат $\mathbb{A}$ ініційований граматикою G або її структурою $C(G)$. Нехай завдані граматична структура $AC = \langle V, \{\to^2\}, \Lambda \rangle$ і автомат $\mathbb{A} = \langle A, Z, Z_0, Z_1, \varphi \rangle$ встановимо між ними відношення за схемою:

а) термінальний алфавіт словника V структури приймемо за вхідний алфавіт A автомату;

б) не термінальний алфавіт словника V граматичної структури приймемо за множину станів Z автомату;

в) множині початкових станів Z_0 автомату будуть відповідати не термінали лівих частин продукцій аксіом початку автоматної структури;

г) за множину заключних станів Z_1 автомату приймаються не термінали аксіом виводу граматичної структури;

д) продукціям $p_i : \alpha_i \to b_i\beta_i$ аксіоматики структури ставиться у однозначну відповідність бінарне відношення, яке буде визначати

функцію переходів $\varphi:\alpha_i \xrightarrow{b_i} \beta_i$, наприклад, у вигляді орієнтованої навантаженої символом b_i дуги графу переходів.

Наведенні правила зв'язку між автоматною граматичною структурою і не детермінованим автоматом є однозначними і тому конкретні структури граматик породжують ті ж мови, котрі проводять ініційовані автомати, тобто $AL(G) = L(\mathbb{A})$.

Приклад 5.3. Нехай завдана автоматна граматика

$$G = \langle \{a,b,c\}, \{\alpha,\beta,\gamma,\sigma\}, \{\gamma,\sigma\}, \{\gamma \to c\alpha, \sigma \to a\alpha, \alpha \to b4\beta, \beta \to c\beta, \beta \to c4, \beta \to a\} \rangle, \tag{5.37}$$

побудуємо відповідний їй не детермінований автомат.

▷Аналіз граматики і зокрема тупікових виводів дає можливість визначити множину станів автомату $Z = \{\alpha,\beta,\gamma,\sigma,\delta,\lambda\}$, з яких γ та σ є початкові і δ та λ заключні стани. Отже, формально маємо автомат $\mathbb{A} = \langle \{a,b,c\}, \{\alpha,\beta,\gamma,\sigma,\delta,\lambda\}, \{\gamma,\sigma\}, \{\delta,\lambda\}, \varphi \rangle$ з графічним представленням функції переходів.

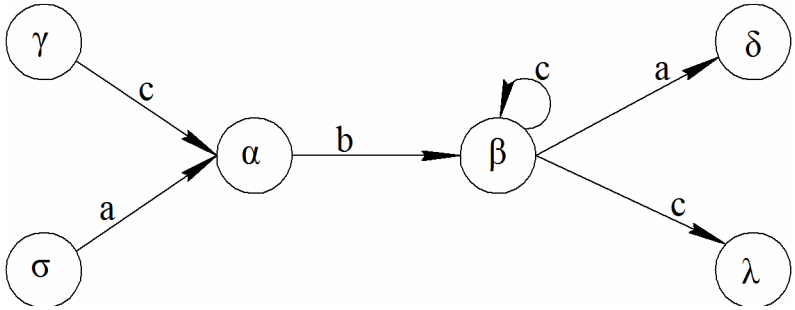


Рис. 5.4. Граф переходів автомату, який відповідає граматиці (5.37)

Наведений на рис. 5.4 граф переходів можна спростити за допомогою правил виключення на графах (див. розд. 3), ввівши один початковий σ і один заключний δ стани (рис. 5.5).

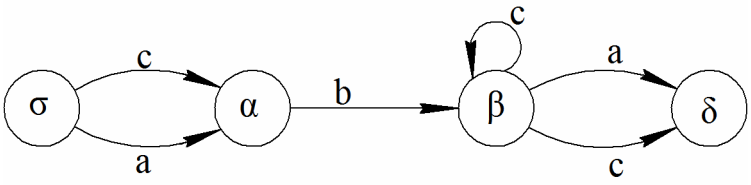


Рис. 5.5. Спрощений граф графу рис. 5.4

Автомат, граф переходів якого зображено на рис. 5.5, відповідає структурі граматики з аксіоматикою

$$\Lambda = \begin{cases} \textit{аксіоми початку}: \\ \sigma \to c\alpha, \\ \sigma \to a\alpha; \\ \alpha \to b\beta; \;\; \beta \to c; \\ \beta \to c\beta; \;\; \beta \to a. \end{cases}$$

Граматична структура з наведеною аксіоматикою породжує складну мову

$$\left\{\{abc^n\} \cup \{abc^k a\} \cup \{cbc^m\} \cup \{cbc^s a\};\ n,k,m,s \in \mathbb{N}_0\right\}. \qquad (5.38)$$

Можна перевірити, що ця ж мова проводиться побудованим автоматом. ◁

Зрозуміло, що автоматній граматиці та її структурі може відповідати ініційований і детермінований автомат (див. п. 4.2.3). Мови, які проводяться детермінованим автоматом називаються *регулярними.* Клас регулярних мов позначимо RL. З'ясуємо, як співвідносяться клас регулярних мов RL і клас автоматних мов AL. З цього приводу існує теорема [3, 16].

Теорема 5.8. Класи мов AL і RL співпадають.

▷ Доведення теореми конструктивно просте. Нехай довільна автоматна мова $L_i \in AL$, тоді існує структура $C(G_i)$, за якою породжується мова L_i. Структурі $C(G_i)$ поставимо у відповідність ініційований автомат $\mathbb{A}_i$, який проводить мову L_i. За теоремою 4.1 для автомату $\mathbb{A}_i$ існує йому функціонально еквівалентний детермінований автомат, тобто мова L_i регулярна і $L_i \in RL$. Отже, маємо включення $AL \subseteq RL$. Аналогічно доводиться включення $RL \subseteq AL$. Таким чином за отриманими включеннями $AL = RL$. ◁

Наведемо деякі рекомендації, якими слід користуватися при приведені недетермінованих автоматів до еквівалентних детермінованих автоматів.

При наявності у недетермінованому автоматі декількох початкових станів їх слід об'єднати в один, для цього можна скористатися правилами виключення альтернативних дуг на графові переходів (див. розділ 3).

а) Якщо в графі переходів існує хоча б одна вершина α_k, з якої виходить менша кількість дуг, ніж розмір вхідного алфавіту автомату, тоді необхідно ввести тупикову вершину ω і з'єднати кортеж (α_k, ω) паралельними дугами, позначивши їх символами, яких не немає серед символів вхідного алфавіту.

б) Нехай в графові переходів є вершина, з якої виходять більше, ніж одна дуга з однаковими позначками. Якщо одна з дуг є петля, тоді її необхідно перенести на суміжну по дузі вершину з тією ж позначкою, або в іншому випадку зоставити петлю та перенести вихідні лінійні дуги в проміжні додаткові вершини (з новими позначками). При необхідності слід з'єднати вершину з петлею і нові вершини з'єднати порожніми дугами.

Скористуємося цими рекомендаціями у розглянутому прикладі 5.3.

Приклад 5.4. Доведемо, що мова (5.38) є регулярною.

▹У прикладі 5.3 показано, що мова (5.38) проводиться не детермінованим автоматом з графом переходів представленим на рис. 5.4. Початкові стани цього автомату зводяться до одного, як показано на рис. 5.5. З вершини β цього графу переходів виходе дуга і петля з однаковою позначкою c тому, за рекомендацією 3 зоставимо петлю на вершині β. На шляху від вершини β до заключної вершини δ введемо допоміжною вершину φ та навантажимо дугу (β, φ) порожнім символом ε і з'єднаємо вершину φ з заключною – δ дугами з позначками a і c.

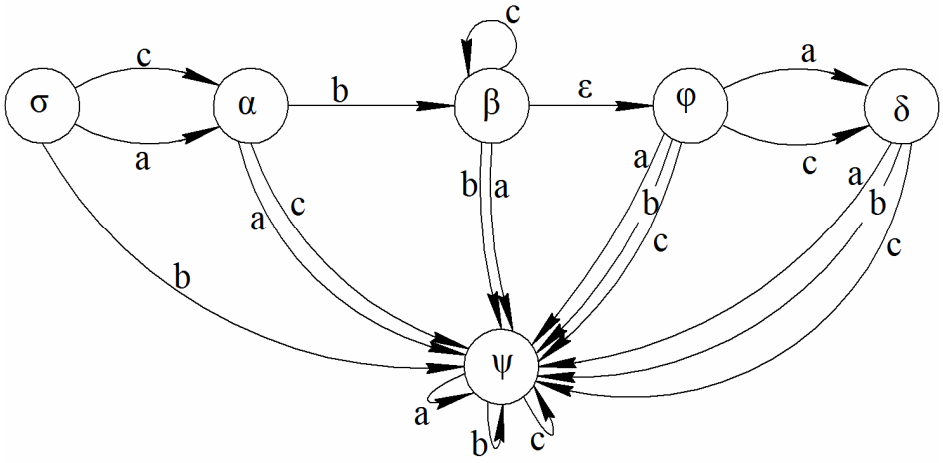


Рис. 5.6. Детермінований граф переходів

Крім того введемо тупикову вершину ψ, на яку, за рекомендацією 2, «скинемо» необхідні дуги, щоб автомат став детермінованим. Та-

ким чином отримаємо граф переходів (рис. 5.6) детермінованого автомату. Для спрощеного представлення графу на рис. 5.6 введені позначки дуг типу (a,b,c), що відповідає трьом дугам, які з'єднують ті ж самі вершини з відповідними позначками. Цьому детермінованому автоматові відповідає граматична структура з аксіоматикою:

$$\Lambda = \begin{cases} \textit{аксіоми початку}: \sigma \to a\alpha,\ \sigma \to c\alpha; \\ \alpha \to b\beta,\ \beta \to c\beta,\ \beta \to \varphi; \\ \textit{аксіоми виводу}: \varphi \to a,\ \varphi \to c. \end{cases} \quad \triangleleft$$

Наведеними класами породжуючої граматичної структури не вичерпується все різноманіття породжуючих класів. Досить широкий спектр породжуючих предметних граматик наведено в інженерному довіднику [12].

Існують також інші класи як детермінованих так і не детермінованих чітких і розмитих граматик.

5.9. Підсумковий коментар

В цьому розділі досить детально розглянуті поняття формальних породжуючих граматик Хомського і конструктивних граматичних структур як основ будови мов програмування, теорії трансляторів та іншого. Граматичні структури взагалі не задають алгоритми, тому, що граматичні правила альтернативні і при відповідних умовах може бути застосовано те чи інше альтернативне правило. Без альтернативним є початкове правило (аксіома початку виводу об'єктів мови).

Слід зауважити, що граматична структура є зовнішньою по відношенню до породженої нею мови, в тому розумінні, що ця ж мова може бути породжена і в іншій структурі. Виходячи з цього запропонована схема алгоритму для відтворення граматичної структури за утворюючими зразками мови.

При завданні штучної мови скінченим чином синтаксичних продукцій (сучасні мови програмування мають сотні синтаксичних правил формування їх фрагментів), деякі необхідні для неї об'єкти відсутні , наприклад, елементи метамови, абстрактні семантичні категорії операторів, структур даних тощо. Введення таких абстрактних категорій зручно робити в предметних конструктивних структурах

запропонованих у розділі. Раніше було показано, що граматичні правила мов програмування можна задавати синтаксичними діаграмами, у цьому розділі запропонована конструктивна граматична структура з нотаціями Бекуса – Наура. За цією структурою можна проводити моделювання штучних граматик, використовуючи апарат систем символьних рівнянь.

Класично, в граматичних правилах застосовується одна операція послідовної підстановки. Але розвиток граматик і граматичних структур дозволив збільшити їх мовну потужність, завдяки ускладненню застосування правил підстановки.

Розглянуті граматики і їх конструктивні структури є лише вступом до будівлі граматичних конструкцій. Зараз складно перелічити всі аспекти граматик, назвемо деякі з них: програмні, графічні, атрибутивні, гібридні та інші [2, 6, 8, 10 – 12, 15, 19, 20].

Література до розділу 5

1. *Алферова З. В.* Теория алгоритмов. – М. Статистика, 1973. – 164 с.
2. *Ахо А.* Индексные грамматики – расширение контекстно-свободных грамматик. // Языки и автоматы. – М.: Мир, – 1975. – С. 130-165.
3. *Гинзбург С.* Математическая теория контекстно-свободных языков. М.: Мир, 1970. – 328 с.
4. *Гладкий А. В.* Формальные грамматики и языки. М.: Наука, 1973. – 368 с.
5. Енциклопедія кібернетики, том 1. – К.: Головна редакція УРЕ, 1973. – 584 с.
6. *Ільман В. М., Скалозуб В. В.* Інтервальні об’єкти та їх граматичні структури // Вісник Дніпропетровського національного університету імені академіка В. Лазаряна. –2010 – вип. 29. – С. 131-144.
7. *Ільман В. М., Шинкаренко В. І.* Структурний підхід до проблеми відтворення граматик // Проблемы программирования. –2007 – № 1. – С. 5-16.
8. *Лисовик Л. П., Карнаух Т. А.* Об одном методе задания фрактальных множеств // Кибернетика и системный анализ. – 2009 – №3. – С. 42-49.

9. *Рейуорд-Смит В. Дж.* Теория формальных языков. Вводный курс. – М.: Радио и связь, 1988. – 128 с.
10. *Розенкранц Д.* Программные грамматики и классы формальных языков. // Сборник переводов по вопросам информационной теории и практики, ВИНИТИ, – 1970. – № 16. – С. 117 – 146.
11. *Саломаа А.* Жемчужины теории формальных языков. – М.: Мир, – 1986. – 159 с.
12. *Фу К.* Структурные методы распознавания образов. – М.: Мир, – 1977. – 318 с.
13. *Хомский Н.* в кн.: Новое в лингвистике, 1962, вып. 2. С. 412 – 527.
14. *Хопкрофт Дж. Е., Ульман Дж. Д.* Формальные языки и автоматы. – М.: Мир, 1982. – 346 с.
15. *Lindenmayer A.* Mathematical models for cellular interaction in development. Parts I and II. // Journal of Theoretical Biology. – 1968. – V. 18. – С. 280 – 315.
16. *Глушков В. М., Цейтлин Г. Е., Ющенко Е. Л.* Алгебра. Языки. Программирование. – К.: Наукова думка, 1989. – 376 с.
17. *Кон П. М.* Универсальная алгебра. – М.: Мир, 1986. – 359 с.
18. *Яблонский С. И., Гаврилов Г. П., Кудрявцев В. Б.* Функции алгебры логики и классы Поста. – М.: Наука, 1966. – 120 с.
19. *Шинкаренко В. И., Ильман В. М.* Конструктивно-продукционные структуры и их грамматические интерпретации. I Обобщенная формальная конструктивно-продукционная структура. // Кибернетика и системный анализ. – 2014. – Том 50, № 5. – С. 8-16.
20. *Шинкаренко В. И., Ильман В. М.* Конструктивно-продукционные структуры и их грамматические интерпретации. II Уточняющие преобразования. // Кибернетика и системный анализ. – 2014. – Том 50, № 6. – С. 15-28.

Розділ 6.
Методи формальних описів та аналізу протоколів. Адміністративне управління інформаційно-обчислювальними мережами

6.1. Мета розділу

Важливість протокольних узгоджень була усвідомлена на ранніх етапах розвитку розподілених систем. Спочатку описання таких угод (специфікації) носили неформальний (словесний) характер. Подібні описання вносили ілюзію простоти при вирішенні виникаючих проблем. Однак, практика швидко переконала розробників у зворотному: суб'єктивна природа сприйняття словесних описань не дозволяє узгодити стандарти, що розробляються; ці тексти носять неоднозначний характер, не мають повноти, не мають формальної основи для аналізу; і вони призводять до несумісності дорогих програмно-технічних виробів і не можуть служити довгостроковою основою для розвитку складних розподілених систем.

Проблема вирішується шляхом *застосування методів формального опису (МФО) протоколів* і сервісів, які повно і однозначно визначають всі аспекти взаємодії. На жаль, не вдалося виробити загальновизнані МФО. Причинами цього є наступне:

– складність, яка викликана громіздкою структурою протокольних об'єктів і комбінаторної складності завдань взаємодії;
– велика різноманітність протоколів – від простих стартстопних протоколів передачі даних до складних протоколів баз даних;
– певні аспекти протоколів важко піддається описанню, наприклад адресації, мультиплексування, управління потоком;
– суперечності між вимогами та описами загальних стандартів, а також описами для реалізації в конкретному програмно-апаратному оточенні.

Враховуючи ці фактори, слід визнати, що, найближчим часом існуюче різноманіття МФО буде збережено.

Перед тим, як описувати протоколи, необхідно розглянути, що саме треба описувати. Відправною точкою при описі протоколів служить багаторівневий характер протокольних архітектур. При проектуванні таких архітектур важливо визначити сервіси, які використовуються, надаються кожним з протокольних рівнів. При визначенні сервісу внутрішня природа і структура рівня неістотні (концепція "чорної скрині"), а увага концентрується на так званій *"спостережній"* поведінці рівня в термінах вхідних і вихідних подій, що відбуваються на його межі (рис. 6.1). Оскільки сервіс за своєю природою є розподіленим, то методи його опису повинні визначати не тільки події, що відбуваються в кожній точці доступу до цього сервісу, але і зв'язок між подіями різних точок доступу. Специфікація сервісу відіграє дуже важливу роль в силу того, що дозволяє виробити формальні вимоги до проектованих протоколів, виконати аналіз протоколів на узгодженість з цими вимогами і служити еталоном для протоколів, що проектуються.

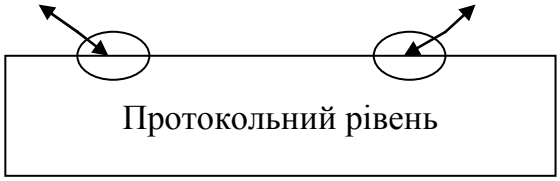


Рис. 6.1. Зовнішнє відображення поведінки протокольного рівня

Після формальної специфікації вимог, що висуваються до протоколу у вигляді сервісу, необхідно більш детально розглянути спосіб виконання цих вимог. Для цього визначаються внутрішня структура протокольного рівня та функції, що ним виконуються. Як правило, протокольний рівень складається з набору протокольних об'єктів, які взаємодіють між собою для узгодження дій. Зробивши опис цих об'єктів, можна повністю визначити протокол. *Внутрішня структура протокольного рівня* представлена на рис. 6.2. З цього рисунка видно, що взаємодії об'єктів виконуються за запитами користувачів у відповідності з протоколом за допомогою сервісу розташованих нижче рівнів. Тому описи об'єктів повинні містити не тільки "протокольні" взаємодії об'єктів одного рівня, але і сервісні взаємодії з об'єктами суміжних рівнів [16] .

Можна виділити два типи формальних специфікацій: специфікацію сервісів, які представляються протокольним рівнем, і специфікацію поведінки протокольних об'єктів у процесі подання сервісів.

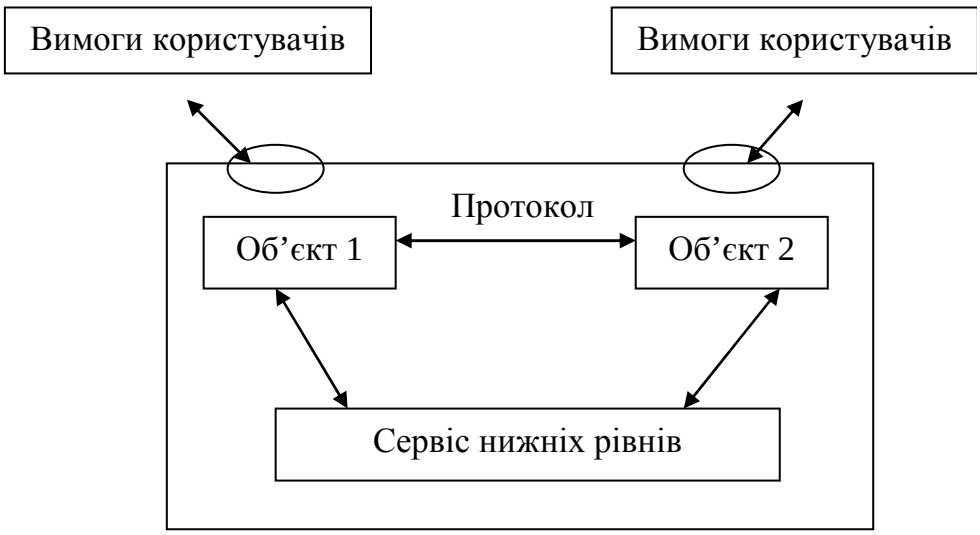


Рис. 6.2. Внутрішня структура протокольного рівня

Незважаючи на різницю цих двох типів специфікацій, до методів їх опису в ISO були вироблені єдині вимоги. МФО повинні забезпечувати:

– несуперечливу, ясну і точну специфікацію;
– основу для визначення повноти представлення протоколів і сервісів, що описуються;
– основу для аналізу протоколів і сервісів відносно коректності, ефективності і т.д .;
– засоби для верифікації того, що протоколи і сервіси відповідають вимогам архітектури, що розробляється;
– базу для визначення узгодженості стандартів один з одним;
– засоби для визначення відповідності реалізацій протоколів стандартам.

Перш ніж перейти до опису конкретних МФО, необхідно розглянути формальні моделі, що лежать в основі цих методів і дозволяють виконати перераховані вище вимоги.

6.2. Моделі для формального описання протоколів

Для успішного вирішення завдання специфікації необхідно вибрати адекватну модель, що лежить в основі МФО і дозволяє задовольнити висунуті до них вимоги. *Численні моделі*, що використовуються для специфікації протоколів і сервісів, можна класифікувати за різними ознаками. Розглянемо дві групи: автоматних моделей та моделей послідовностей.

Автоматні моделі розглядають внутрішній стан об'єкта специфікації та описують всі можливі зміни цього стану при дії на об'єкт. До моделей цього виду відносяться:

- традиційні (синхронні та асинхронні) кінцеві автомати,
- регулярні вирази,
- мережі Петрі,
- розширені автомати,
- формальні граматики,
- алгоритмічні мови.

Моделі послідовностей розглядають тільки поведінку об'єкта, що спостерігається "зовні" (зовнішнє), не допускаючи ніяких припущень щодо його внутрішньої структури. *До таких моделей відносяться*:

- абстрактні типи даних,
- трасові специфікації,
- описи, що використовують часову логіку,
- обчислення взаємодіючих процесів,
- змінні історії.

Хоча обидві групи моделей можуть застосовуватися як для специфікації протоколів, так і для специфікації сервісів, ясно, що природа моделей послідовностей більш придатна для специфікації сервісів, а автоматних моделей – для специфікації протоколів. Найбільш характерні особливості тієї та іншої групи моделей наведені в табл. 6.1.

У таблиці під спільністю розуміється здатність опису задовольнити як можна більшу числу способів реалізації.

Таблиця 6.1

Характерні особливості моделей МФО

Автоматні моделі	**Моделі послідовностей**
Більш добре вивчені	Менш вивчені
Мають меншу спільність	Більш загальні
Більше придатні для отримання безпосередньої реалізації	Менш придатні для отримання безпосередньої реалізації
Наочні і менше абстрактні	Менш наглядні і більш абстрактні
Швидке зростання числа станів	Позбавлені недоліку зростання числа станів
Основний метод аналізу – аналіз досяжних станів	В основі аналізу лежать математичні методи
Більше придатні для опису протоколів	Більше придатні для опису сервісів

6.3. Автоматні моделі

6.3.1. Кінцеві автомати

Специфікації протоколів на основі моделей кінцевих автоматів в даний час використовуються найбільш широко. Формально кінцевий автомат (КА) визначається шістьма об'єктами

$$K_A = \{S, I, O, N, M, S_0\}, \tag{6.1}$$

де S – кінцева множина станів, I – кінцева множина входів, O – кінцева множина виходів, $N : I \times S \to S$ – функція переходів, $M : I \times S \to O$ – функція виходів, S_0 – початковий стан.

Функція N і M описують поведінку автомата, тобто якщо в деякому поточному стані $s_i \in S$ на вході автомата з'являється повідомлення $i \in I$, то функція переходів визначає новий стан автоматів $s_i \in S$, а функція виходів - вихідне повідомлення $o_i \in O$.

Для опису КА використовуються різні способи, однак найбільш широко поширені діаграми станів і таблиці рішень. На рис. 6.3 представлені діаграми станів КА, які описують поведінку передавача і приймача відповідно добре відомого протоколу АВР (протокол з нумерацією по модулю 2).

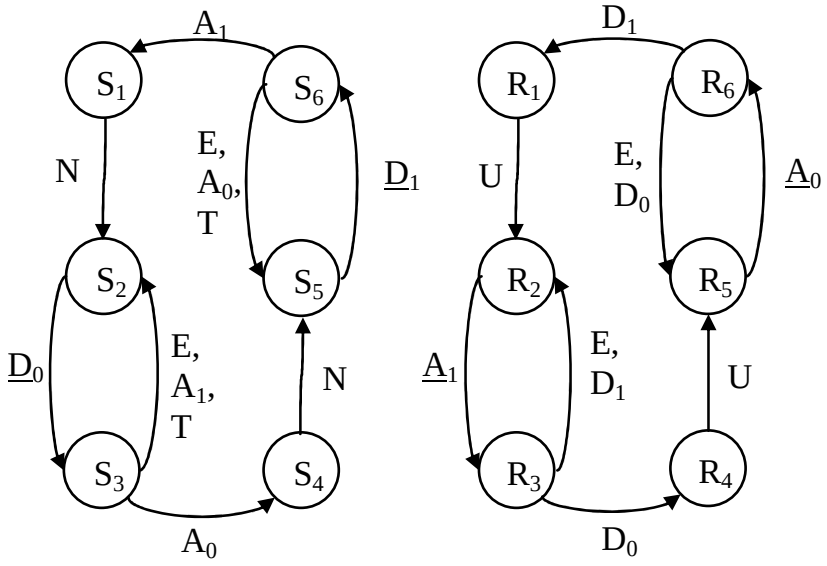


Рис. 6.3. Діаграма станів протоколу АВР

Діаграма станів представляє собою різновид спрямованого графа, множина вершин якого відповідає множині станів, а множина дуг –

множині переходів. Дуги графа позначаються відповідними вхідними та вихідними повідомленнями. Передача повідомлень між КА моделюється чергами, які в окремому випадку можуть бути виродженими (так звана абстракція "пустого" середовища [11]). В останньому випадку переходи, що пов'язані з передачею повідомлень в одному КА (символи підкреслені) і прийомом цього повідомлення в іншому КА, безпосередньо зв'язані один з одним. Крім того, є можливість визначити спеціальний тип вхідної дії, не пов'язаної з передачею повідомлень від одного КА до іншого і зветься "внутрішня" подія. *Внутрішня подія* (підготовка нового повідомлення N і обробка отриманого повідомлення U на рис 6.3) відображають вплив на автомат зовнішніх дій, не включених в специфікацію протоколу. Для рис. 6.3 справедлива наступна семантика позначень: N – надходження нового повідомлення, D_i – передача повідомлення з номером I, E – прийом спотвореного повідомлення, A_i – отримання підтвердження про прийом повідомлення з номером i, T – сигнал завершення заданого інтервалу часу (тайм-аут), U – обробка отриманого повідомлення, A_i – передача підтвердження про прийом повідомлення з номером i, D_i – прийом повідомлення з номером i.

Таблиця рішень, що відповідає діаграмі станів передавача рис. 6.3, представлена в табл. 6.2. Стовпці таблиці відповідають входам $i \in I$, а рядки – станам $s_i \in S$. Елементами таблиці є пари (s, o), де s_i – новий стан і $o_i \in O$ – вихід. Прочерк у відповідному елементі таблиці означає відсутність вихідного повідомлення або те, що прийом даного вхідного повідомлення не передбачений. Зірочкою позначений пустий вхідний сигнал.

Початковим станом передавача вважається s_1, а приймача – R_3. В цьому стані передавач очікує надходження нового повідомлення для передачі, а приймач обробив всі отримані раніше повідомлення.

Таблиця 6.2

Таблиця рішень, що відповідає діаграмі станів передавача

Стан	Тип вхідного сигналу					
	N	E	A_1	T	A_0	*
s_1	s_2 , -	- , -	- , -	- , -	- , -	- , -
s_2	- , -	- , -	- , -	- , -	- , -	s_3 , D_0
s_3	- , -	s_2 , -	s_2 , -	s_2 , -	s_4 , -	- , -
s_4	s_5 , -	- , -	- , -	- , -	- , -	- , -
s_5	- , -	- , -	- , -	- , -	- , -	s_6 , D_1
s_6	- , -	s_5 , -	s_1 , -	s_5 , -	s_5 , -	- , -

Даний метод специфікації має наочність і широко використовується. Але із зростанням числа станів КА використання цієї моделі стає все більш утрудненим. Наприклад, при описі за допомогою КА протоколів передачі даних можливому значенню номера повідомлення, що передається або приймається, повинен відповідати окремий стан КА. Тому КА найбільш зручні для опису структури взаємодій протоколів (наприклад, таких аспектів, як встановлення або розрив з'єднання, відновлення після помилок), коли кількість станів відносно незначна.

6.3.2. Розширені кінцеві автомати

Для вирішення проблеми розмірності класичного КА запропонована модель розширеного кінцевого автомата (РКА), в якій використовується принцип декомпозиції.

Інформація, якою обмінюються протокольні об'єкти, може бути представлена у вигляді <код операції> <параметри>.

<Код операції> визначає вид запиту користувача або повідомлення, а <параметри> використовуються для передачі додаткової інформації відповідно до кожного конкретного виду. Визначимо множину входів I наступним чином:

$I = \{$<код операції> <параметри>$\}$,

$I_1 = \{$<код операції>$\}$, $I_2 = \{$<параметри>$\}$, $I \subset I_1 \times I_2$.

Важливо відзначити, що потужність множини I_1 набагато менше потужності множини I_2. Тому будь-якому входу з множини I_1 протиставимо стан або множину станів кінцевого автомата, який назвемо базовим, а <параметрам> – множина контекстних змінних. Таким чином, залежність параметрів від коду операції відобразиться в залежність контекстних змінних від станів базового автомата. Більш формально це можна виразити наступним чином. Нехай для кінцевої множини станів справедливо $s \in S_1 \times S_2$, де S_1 – множина значень <код операції>, S_2 – множина значень <параметри>. *Тоді кінцевий автомат є п'ятірка об'єктів:*

$$<S_1 \times S_2, I_1 \times I_2, Y, P, Q>, \qquad (6.2)$$

де Y – множина вхідних символів; $P[S_1 \times I_1]: S_2 \times I_2 \to S_2$ – функція переходів; $Q[S_1 \times I_1]: S_2 \times I_2 \to Y$ – функція виходів.

Вирази показують, що відображення $S_2 \times I_2$ в S_2 і Y залежить від стану базового і вхідних символів множини <код операції>.

Базовий автомат – п'ятірка об'єктів

$$<S_1 \times S_2, Y, O, F, G>, \tag{6.3}$$

де $F: Y \times S_1 \to S_1$; $G: Y \times S_1 \times S_2 \to O \to O$.

Декартовий добуток $O \times O$ вводиться для функції G, тому що вхідний елемент з Y може породити два виходи.

Щоб зрозуміти, як вихідна функція G залежить від множини S, структуруємо кожний елемент вихідної множини O як і для випадку вхідної множини: $O =$<код операції / відповіді> <параметри>, $O_1 =$ {<код операції / відповіді>}, $O_2 =$ {<параметри>}.

Тоді вихідна множина O буде підмножиною декартового добутку $O_1 \times O_2$.

Для того щоб конструювати елементи множини $O_2 =$ {<параметри>}, необхідно знати значення деяких контекстних змінних S_2.

6.3.3. Мережі Петрі

Більш широкими можливостями, ніж діаграми станів, в описі структури і поведінці складних систем мають мережі Петрі [3]. Для специфікації протоколів використовуються звичайні мережі Петрі і різні їх розширення (інтерпретації).

Звичайна мережа Петрі (МП) задається п'ятіркою об'єктів:

$$I =< P, T, F, B, M_0 >, \tag{6.4}$$

де P – кінцева множина позицій, T – кінцева множина переходів, $F: P \times T \to N$ – функція входів, $B: P \times T \to N$ – функція виходів, $M_2: P \to N$ – початкова розмітка, $N: S_2 \times I_2 \to S_2 \geq 0$ – ціле число.

Найбільш зручно представити МП у вигляді дводольного мультиграфа з множиною вершин $P \bigcup T$. Елемент з множини P зображується кружком, а елемент з множиною T – рисою. Елементи різних множин *можуть з'єднуватися направленими дугами*. Якщо з позиції p до переходу t веде k дуг, то $F\ (p,t) = k$, і якщо з переходу t до позиції p веде l дуг, то $B\ (t, p) = 1$. Кожна позиція МП характеризується певним числом міток і визначається функцією $M: P \times N$, яка називається розміткою. Графічно розмітка представляється як розподіл міток в позиціях мережі, зображених точками.

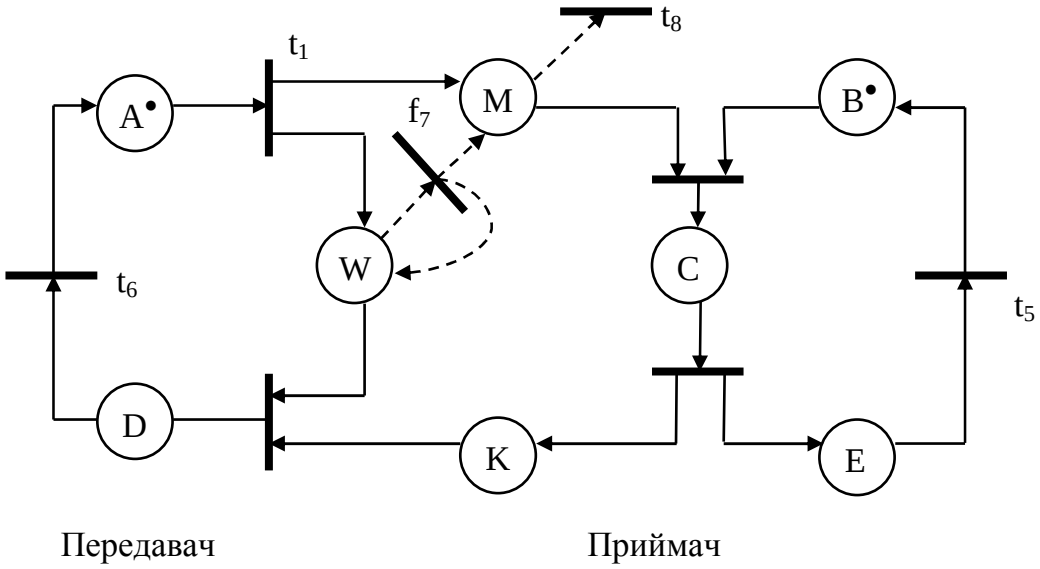


Рис. 6.4. Мережа Петрі для протоколу АВР

Функціонування МП визначається правилами виконання переходів. Перехід t_i зветься збудженим, якщо для всіх позицій, що мають дуги, які входять в даний перехід, виконується $M(p) - F(p,t_i) \geq 0$, де $M(p)$ – розмітка позиції р, тобто кожна вхідна позиція містить не менше однієї мітки. Перехід може бути збудженим протягом довільного, але кінцевого часу. Потім він або спрацьовує, або збудження з нього знімається (при спрацьовуванні іншого переходу).

У результаті спрацювання переходу t_i розмітка змінюється за правилом $p \in P((M'(p) - F(p,t_1) + B(p,t_1)),\ t_i, t_i \in T$. Тобто з кожної вхідної позиції переходу віднімають по одній мітці, а кожній вихідний позиції додається по одній мітці. В кожний момент часу може спрацювати тільки один перехід. Якщо ж одночасно збуджені декілька переходів, то спрацьовує один довільний.

Стан МП в кожний момент часу визначається її розміткою. Якщо задана початкова розмітка і хоча б один перехід є збудженим, то в мережі починається рух міток. *Говорять, що розмітка* M_0 породжує послідовність спрацьовувань переходів $\sigma = t_1, t_2, \ldots, t_k$, і це приводить до нової розмітки $M \in M(M_0)$. Множина $M(M_0)$ представляє собою множину розміток, що утворюються з M_0 в результаті спрацьовування різних послідовностей переходів, і називається множиною досяжних з M_0 розміток. МП можна представити графом досяжних розміток, який містить всі можливі розмітки і переходи між ними.

На графі досяжних розміток (рис. 6.5) позиції визначають стан передавача P_1 і приймача P_2: A – передавач готовий передати повідом-

лення; B – приймач готовий прийняти повідомлення; C – повідомлення прийнято приймачем; E – підготовка приймача до прийому повідомлення; D – підтвердження отримано передавачем; K – підтвердження передано приймачем; M – повідомлення передано передавачем; W – очікування підтвердження передавачем. В початковому стані $M(A)=1$, $M(B)=1$ і всі інші позиції, що залишилися пусті. Це означає, що P_1 готовий видати повідомлення. Після того, як спрацьовує перехід t_1, відбувається передача повідомлення $M(M)=1$ і передавач переходить в стан очікування прибуття підтвердження $M(W)=1$. Якщо приймач P_2 готовий прийняти повідомлення $M(B)=1$, то спрацьовує перехід t_2. $M(C)=1$ означає, що повідомлення прийнято. Після цього спрацьовує перехід t_4, котрий проставляє мітки до вузлів E і K; $M(E)=1$ означає оповіщення приймача P_2 про те, що повідомлення прийнято, а $M(K)=1$ – що передано підтвердження. Останнім спрацює перехід t_3, в результаті чого передавач P_1 буде сповіщений про завершення передачі повідомлення $M(D)=1$. Якщо тепер P_1, підготувавши нове повідомлення, розмістіть мітку в позицію A, а P_2, обробивши повідомлення, помістить мітку в позицію B, то модель почне новий цикл функціонування.

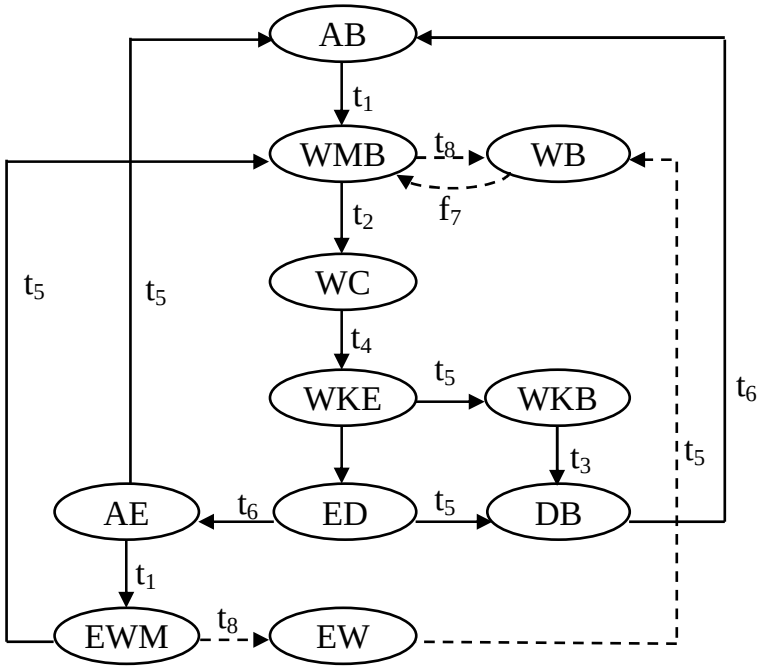


Рис. 6.5. Граф досяжних розміток, який відповідає мережам Петрі протоколу АВР

Часові мережі Петрі. *Звичайні мережі Петрі* не містять характеристики про час в явному вигляді. Тому з їх допомогою не можна описати тривалість виконуваних протокольним об'єктом операцій, наприклад тайм-аутів, обробки повідомлень і т.д. Таку можливість забезпечують часові мережі Петрі (ЧМП) [18], в яких кожному переходу t_i ставляться у відповідність два інтервали часу, а саме $\min T_i$ і $\max T_i$, що задовольняють умовам:

– перехід може спрацювати не раніше, ніж через інтервал часу $\min T_i$ після його збудження;

– перехід t_i повинен спрацювати не пізніше ніж через інтервал часу $\max T_i$ після його збудження.

Іншими словами, з кожним переходом зв'язується деяка часова затримка. Якщо $\min T_i = 0$, а $\max T_i = \infty$, то ЧМП перетворюється на звичайну МП.

Розглянемо мережу Петрі на рис. 6.4. Втрата повідомлень в передавальному середовищі моделюється шляхом видалення мітки з позиції M. Для ініціації повторної передачі повідомлень, втрачених в середовищі, вводиться перехід t_7, що означає спрацьовування таймера. У наступному розділі буде показана методика вибору значень часових характеристик переходу t_7 на основі відомих часових характеристик інших переходів з тим, щоб забезпечити правильне функціонування моделі. Тут відзначимо лише, що повинно виконуватися співвідношення

$$\min T_7 > \max T_2 + \max T_3 + \max T_4 .$$

Розширення мережі Петрі. В рамках загальної теорії мережі Петрі отримали розвиток різні підходи до опису та аналізу складних обчислювальних систем. *Результатом стала поява нових* інтерпретацій теорії мереж, які дозволяють компактно описувати складні системи і в той же час допускають більшість існуючих методів аналізу.

У відповідності зі ступенем допустимої компактності опису моделі виділяються наступні класи мереж Петрі [4]:

– мережі типу "позиція / перехід" (МП, ЧМП);

– розфарбовані мережі Петрі (РМП);

– мережі типу "предикат / перехід" (МП-ПП);

– мережі типу "предикат / дія" (МП-ПД).

У розфарбованих мережах Петрі кожній мітці ставиться у відповідність елемент деякої множини, зокрема множина кольорів. На від-

міну від звичайної мережі Петрі, де мітки відрізняються, мітки в РМП мають певну індивідуальність. Для кожного переходу визначається множина допустимих поєднань кольорів вхідних міток, при яких перехід вважається збудженим і може спрацювати. Кожному такому поєднанню кольорів міток ставиться у відповідність поєднання кольорів міток вихідних місць, яке встановлюється в результаті спрацьовування переходу. Більш точне визначення можна знайти в [4].

Позиція P_1 (рис. 6.6, а) моделює лічильник, який містить номер очікуваного пакету. Роль кольорів в цьому випадку грають цілі числа, циклічно приймаючі значення від 0 до 4. Позиція P_2 моделює буфер вхідного повідомлення, що має номер від 0 до 4. Мітка в позиції P_3 не має кольорів і аналогічна позиції звичайної МП. Перехід t збуджений, якщо в позиціях P_1 і P_2 знаходяться мітки з однаковим номером i. У результаті спрацьовування переходу з позицій P_1 і P_2 видаляють мітки з номером i, а в позицію P_1 повертається мітка з номером $i+1 \,(\mathrm{mod}\, 4)$, тобто з номером наступного очікуваного повідомлення. У позицію P_3 поміщається мітка, що сигналізує про надходження повідомлень. Таким чином, перехід t спрацьовує тільки у випадку рівності номерів очікуваного і прийнятого повідомлень.

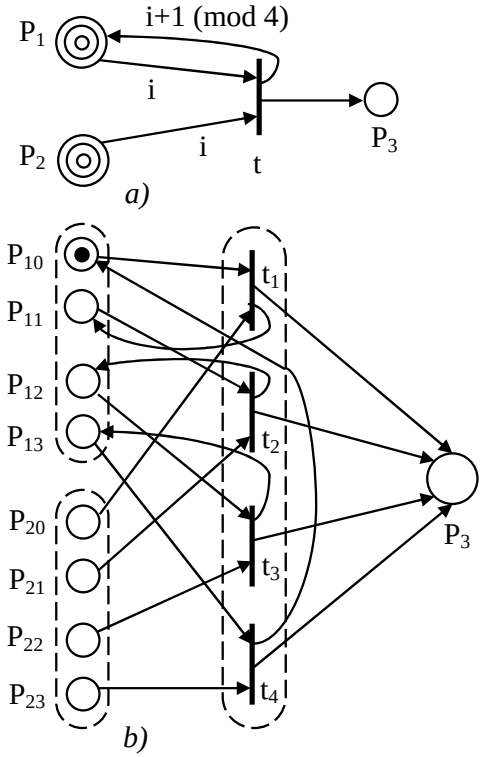


Рис. 6.6. Фрагмент РМП, який моделює прийом пакету даних (а), і еквівалентна їй звичайна МП (б)

Якщо кількість кольорів РМП обмежене, то така мережа за своєю виразною потужністю *еквівалентна звичайній мережі Петрі*. Існує формальна процедура переводу РМП в еквівалентну МП [3] (рис. 6.6, б). З рисунку видно, що РМП дозволяє компактно представити складні і громіздкі системи.

Мережі Петрі типу "предикат / перехід" (МП-ПП) дозволяють досягти ще більшої компактності представлення складних систем, ніж РМП. Це можна досягнути за рахунок визначення додаткових відносин між мітками. Для спрацювання переходу необхідно, щоб не тільки у вхідних позиціях знаходилися мітки певних типів, але і щоб між ними виконувалися задані відношення.

Неформально МП-ПП включає розкрашену мережу Петрі, структури операцій і відношень між мітками, предикати, які можуть бути зв'язані з переходами. Останні задають відношення, які повинні виконуватися, щоб перехід спрацював; предикати описуються в переходах.

Таким чином, щоб перехід спрацював, необхідно:

– відповідність між кольорами міток, що з'являються на вхідних і вихідних дугах переходів (умова спрацьовування РМП);
– предикат, описаний в переході, повинен приймати значення "істина". Мережі типу МП-ПП описані у [6, 7].

Пропонується ще одна графічна модель МП, яка використовує предикати, котра зветься числові мережі Петрі (ЧМП). Відмінність ЧМП від МП-ПП в тому, що в ній додається новий елемент – пам'ять, що складається із змінних. *Над змінними* можуть виконуватися операції читання і запису. Умови збудження переходів визначаються наступним чином:

– для кожної дуги, що входить в перехід, існує умова спрацювання, котра може висувати вимоги як до кількості міток в позиції, так і до їх значень;
– для кожного переходу вводиться умова спрацювання для змінних пам'яті, які визначають допустимі співвідношення між ними;
– перехід збуджений, коли для кожної вхідної дуги виконується умова спрацювання, а також виконується умова спрацювання для змінних.

В результаті спрацювання збудженого переходу відбувається наступне:

– для кожної вхідної і вихідної дуги переходу визначаються мітки, які в результаті спрацьовування повинні бути видалені з вхідних позицій і поміщені у вихідні;

– з переходом зв'язується процедура зміни змінних пам'яті;
– спрацювання переходу, що включає видалення і розміщення міток, а також зміну змінних, вважається неподільною операцією.

За властивостями виразності ЧМП еквівалентні МП-ПП.

Мережі Петрі типу "предикат / дія" (МП-ПД). В основі їх лежить гібридна модель Келлера [19], яка описує як структуру управління паралельних програм, так і операції з даними. Структура управління задається звичайною МП. Для опису іншій частини моделі вводиться вектор змінних ξ. До кожного переходу мережі прикріплюється вираз виду

$$when\, Pt(\xi)\ do\, \xi \leftarrow Ft(\xi),$$

де Pt – предикат і Ft – функція, визначена на множині значень вектора змінних ξ. Перехід вважається збудженим, якщо він збуджений в звичайному сенсі і, крім того, відповідний предикат приймає значення істини. При спрацьовуванні переходу Ft змінює значення вектора змінних ξ.

Використання МП-ПД для специфікації протоколів описано в [11, 19].

6.3.4. Формальні граматики

Протокол можна визначити як множину правил, що описують послідовність дій під час взаємодії об'єктів протоколу. Якщо позначити всі ці дії як терміни деякої граматики, то протокол можна представити за допомогою деякої мови. «Висловлювання» цієї мови повинні визначати правильні послідовності протокольних дій. Протокольні дії зазвичай оперують з блоками інформації, що передається, ієрархічна структура котрих також може бути описана за допомогою формальної мови.

6.3.5. Регулярні вирази

Протокольна система представляється довільним числом взаємодіючих послідовних процесів. Кожен процес має по одній поштовії скриньки, через котру він приймає повідомлення. В свою чергу, процес може направити повідомлення в поштову скриньку системи, включаю-

чи свою власну. Над поштовими скринями визначені дві операції "передати" і "прийняти". Операція "прийняти" в якості свого результату виробляє саме старе повідомлення поштової скрині, над якою вона виконується, якщо вона не пуста. У випадку пустої скрині операція "прийняти" затримує прибуття повідомлення. Процес може специфікувати максимальний час очікування прибуття повідомлення. Якщо цей час вичерпано, то результат операції "прийняти" полягає у видачі повідомлення, що сповіщає про закінчення часу очікування.

Операція "передати" закінчується після того, як повідомлення доставляється в поштову скрині адресата. Передбачається, що місткість поштових скринь обмежена. Якщо поштова скриня адресата переповнена, то результатом операції "передати" є сигнал про аварійне завершення (втрата повідомлення), інакше виробляється сигнал про нормальне завершення.

Передбачається, що будь-яке повідомлення унікально визначає процеси джерела і призначення. Так, якщо процес A передає повідомлення m процесу B, то більше в системі немає процесів, що використовують повідомлення з ім'ям m. На практиці таке положення може бути досягнуто *додаванням до імені повідомлення імен процесів джерела і призначення*.

6.3.6. Мови програмування

Моделі, що будуються на використанні мов програмування, розглядають протокол як один з типів алгоритмів, для опису якого використовується мова. В залежності від того, наскільки високі рівень і абстрактність використованої мови, цей підхід до специфікації може вирішувати також проблеми реалізації протоколу.

Основною перевагою даного підходу є вирішення проблеми розмірності, що досягається використанням змінних величин і параметрів. Недоліком мовних специфікацій є те, що незначні деталі програми приховують важливі аспекти протоколу. Розвиток мовних специфікацій має два напрями. У відповідності з першим мова розширюється спеціальним класом операторів, що описують поведінку протокольних систем *(часова логіка).* Другий підхід – створення мов специфікацій, в яких крім звичайних мовних конструкцій вводяться спеціальні конструкції, які специфікують протокольні моделі.

6.4. Моделі послідовностей взаємодій

6.4.1. Абстрактні типи даних

Абстрактні типи даних виникли з ідеї об'єднати дані і операції маніпулювання даними. Специфікації з використанням абстрактних типів даних називають іноді *алгебраїчною специфікацією*. Більш формально під алгебраїчною специфікацією розуміється трійка:

$$< S, \Sigma, E >,$$

де S – кінцева множина імен типів; Σ –кінцева множина імен операцій, замкнутих на S; E – множина аксіом, що визначаючих результати операцій над типами, що визначаються.

Привабливою властивістю абстрактних типів є незалежність опису від способу реалізації і можливість використання в процесі аналізу специфікацій автоматизованих систем логічного виводу. *Методи, що базуються* на абстрактних типах, мають ряд обмежень, пов'язаних з описом паралельності і композиції різних підсистем, а також вимагають багато зусиль для формулювання набору аксіом.

6.4.2. Часова логіка

Часова логіка відноситься до класу модальних логік і дозволяє представити специфіковані об'єкти *у вигляді тверджень над* послідовностями подій. В розглянутих послідовностях можна виділити поточне положення і зробити висновки щодо властивостей послідовностей в майбутньому. Твердження щодо властивостей мають вираз у вигляді формул часової логіки. Властивості послідовності, які виконуються в будь-який момент часу, називаються аксіомами часової логіки.

Приклади специфікації протоколів і сервісів можна створити за допомогою аксіом часової логіки можна знайти в [13]. *Аналіз таких специфікацій* дозволяє довести властивість живучості та безпеки.

6.4.3. Обчислення взаємодіючих процесів (ОВП)

Обчислення взаємодіючих процесів було розроблено Мільнером і базується на невеликій множині примітивних понять. Основним з них

є взаємодія. Взаємодія відбувається між агентами (процесами). Для побудови нових видів агентів з вже існуючих визначені комбінатори (оператори). Комбінатори вибрані так, щоб відношення між вихідними і утвореними з них агентами можна було трактувати в математичному сенсі.

Теорія ОВП була використана при розробці специфікації LOTOS, а властивість еквівалентності, що спостерігається, використана при верифікації протоколів.

6.5. Формальні методи специфікації протоколу

6.5.1. Попередні зауваження

Розглянуті в попередньому розділі методи формальної специфікації дозволяють виконати аналіз протоколів і наданих ними сервісів. В даному розділі будуть розглянуті методи аналізу, пов'язані не з оцінкою якісних характеристик виконання протоколів, а з їх логічної правильністю.

Для оцінки якісних характеристик можна використовувати добре відомі методи аналітичного або імітаційного моделювання або методи "передбачення" характеристик виконання[9].

Всі методи аналізу логічної правильності розділяються на дві великі групи – методи аналізу коректності та методи верифікації.

Під методами аналізу коректності розуміються загальні властивості, які повинні мати протоколи та сервіси, незалежно від функцій, які ними виконуються. Так, якщо протокол містить можливість блокування *(глухий кут)* при виконанні якої-небудь функції, то він некоректний. *Формальні специфікації* протоколів і сервісів можна аналізувати незалежно один від одного з метою виявлення того, чи виконуються зазначені загальні властивості.

Після того, як аналіз коректності виконаний, потрібно показати, що специфікації протокольних об'єктів спільно зі специфікацією використаного сервісу нижніх рівні відповідають специфікації сервісу, наданого рівнем, на котрому знаходяться ці об'єкти (рис. 6.7). Аналіз такої відповідності носить назву *верифікації*. Подібне застосування терміну "верифікація" співпадає з традиційним використанням його в програмуванні, якщо під специфікацією розуміти специфікацію сервісу, а під способом виконання цих вимог – специфікацію протоколу (протокольних об'єктів).

В даному розділі ми будемо використовувати термін "верифікація" для доказу того, що кінцева програмна реалізація задовольняє специфікації протоколу, так як, з одного боку, процес такого доказу нічим не відрізняється від традиційної верифікації програм і, з другого боку, розвиток систем автоматичної трансляції протокольних специфікацій в реалізації взагалі усуває необхідність у такому вигляді верифікації.

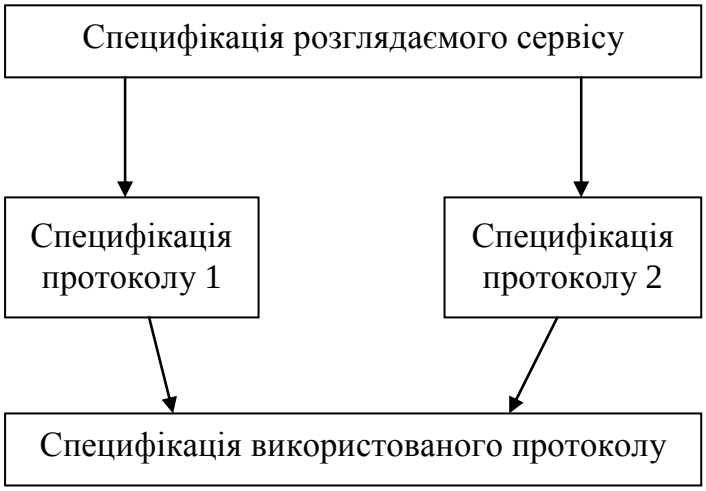


Рис. 6.7. Формальна специфікація рівня

6.5.2. Властивості коректності протоколів

В технології розробки протоколів існує набір властивостей, обов'язковий для будь-яких протоколів. Тому перевірка коректності протоколу повинна забезпечувати доказ того, що специфікація протоколу має ці властивості і вони є наступними:

1) відсутність статичних блокувань. Це означає, що в протоколі не існує такого стану або набору станів, з яких немає переходів в інші стани;

2) повнота, тобто протокол забезпечує реакцію на всі можливі вхідні повідомлення (відсутні помилки не специфікованих прийомів);

3) однозначність відповідності станів, тобто відсутність таких протокольних об'єктів, у яких один стан може співіснувати з декількома різними станами будь-якого іншого об'єкта;

4) відсутність надлишковості, тобто в специфікації протоколу немає повідомлень, які не надходять і невиконуваних дій;

5) обмеженість, котра значить, що під час функціонування протоколу число повідомлень в кожному каналі між протокольними об'єктами не перевищує певного значення, званого ємності каналу;

6) відсутність динамічного блокування. Це значить, що в прото-

колі відсутній безкінечний цикл функціонування, при якому не виконується корисна робота. Розрізняють динамічні блокування, вихід з яких логічно неможливий, і динамічні блокування, які є наслідком певних часових характеристик протоколу, наприклад темпу обміну повідомленнями.

7) завершуваність (розвиток), тобто протокол завжди досягає кінцевого (термінального) стану. Для циклічних протоколів це властивість дещо змінюється. Ці протоколи повинні мати властивість розвитку, котра полягає у тому, що протокол досягає свого початкового стану;

8) самосинхронізація (відновлення після ненормальної ситуації). Ця властивість має на увазі, що після виникнення ненормальної ситуації протокол за кінцевий час відновить своє коректне функціонування.

Розроблені різні класифікації протокольних властивостей. Можна виділити декілька основних груп властивостей коректності протоколів, всередині кожної з яких протокол розглядається з певної точки зору. В свою чергу, кожна з груп розділяється на дві альтернативні категорії. Приклад такого поділу властивостей:

– синтаксичні та семантичні;

– статичні і динамічні;

– часткову коректність і завершуваність;

– безпека і живість.

Синтаксичні властивості визначають логічну структуру обміну повідомленнями (властивості 1...5). Семантичні властивості пов'язані із змістовною правильністю процесу функціонування протоколу. Розрізняють властивості слабкої семантики, які пов'язані з ефективним розвитком протоколу (властивість 7), і властивості строгої семантики, які пов’язані з конкретним призначенням протоколу (властивість 8).

До статичних властивостей відносяться властивості 1 ... 4, а до динамічних властивостей 6 ... 8.

Часткова коректність значить, що протокол має певний набір необхідних властивостей за умови, що в процесі функціонування він досягає заданого термінального стану (або початкового стану у випадку циклічних протоколів). Ця властивість означає лише умовну можливість виконання необхідних властивостей. Повна коректність протоколу вимагає додатково властивості 7.

Безпека значить, що протокол у процесі функціонування не потрапить в неприпустимий стан, і охоплює часткову коректність і властивість 8.

Живість має на увазі, що протокол зрештою досягне бажаного стану, і включає властивості 6, 7 і 8.

6.6. Методи аналізу коректності

6.6.1. Метод аналізу досяжних станів

Метод аналізу досяжних станів оснований на вичерпному переборі можливих взаємодій протокольних автоматів. При цьому розглядається глобальний стан системи, котрий визначається як комбінація станів взаємодіючих протокольних автоматів і стану нижніх рівнів (передавального середовища). Легко помітити, що у випадку протокольних автоматів з однаковим числом станів число глобальних станів системи дорівнює квадрату числа станів протокольного автомата, помноженому на число станів передавального середовища.

Перебір починається з деякого початкового стану, із якого під впливом можливих інтерфейсних, протокольних або таймерних подій створюються переходи до інших глобальних станів. Ці нові стани розглядаються далі як вихідні, і процес рекурсивно продовжується до тих пір, поки не будуть розглянуті всі можливі глобальні стани системи.

Аналіз отриманого графа досяжних глобальних станів дозволяє перевірити загальні властивості протоколу, що випливають безпосередньо зі структури графа досяжності (властивості 1 ... 6).

Зазвичай даний метод аналізу застосовується для дослідження протоколів, специфікація яких заснована на кінцевих або розширених автоматах, мережах Петрі і формальних граматиках. До переваг методу слід віднести зручну графічну форму представлення і досить простий спосіб автоматизації процесу аналізу. Створені на основі цього *методу автоматизовані системи* застосовувались для дослідження реальних протоколів. Основним недоліком методу є швидке зростання числа глобальних станів із зростанням складності протоколів. Відомі декілька різновидів даного підходу: *метод перебору*, *метод діалогових матриць*, *метод фазових діаграм*, *метод "прилеглих" станів* і *метод спільних шляхів*.

Класичний метод перебору. Метод застосовано в моделі взаємодіючих кінцевих автоматів, об'єднаних каналом обмеженої ємності типу FIFO.

Розглянемо N взаємодіючих автоматів. Поточний стан кожного автомата позначимо через $G(I)$, де $I = 1 \ldots N$. Між кожною парою автоматів є два однонаправлених каналів для обміну у двох напрямках.

Позначимо через $K(I, J)$ канал, що зв'язує автомат I з автоматом J. Канали мають такі властивості:

– канал $K(I, J)$ приймає події від автомата I і доставляє їх без зміни послідовності до автомату J;
– час передачі події не визначений і не передбачений;
– кожний канал має максимальну ємність $Cap(I, J)$, що характеризується числом подій, які він може передавати одночасно.

Глобальний стан системи S є множина поточних станів автоматів $G(I)$, $I = 1 \ldots N$, і каналів $K(I, J)$, $I = 1 \ldots N$, $J = 1 \ldots N$, $J \neq I$. *Глобальний стан представляється у вигляді матриці*

$$S = \begin{bmatrix} G(1) & \ldots & K(1,I) & \ldots & K(1,N) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ K(I,1) & \ldots & G(I) & \ldots & K(I,N) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ K(N,1) & \ldots & K(N,I) & \ldots & G(N) \end{bmatrix} \qquad (6.5)$$

Перехід з даного поточного стану S в інший стан S' відбувається після виконання одиночного переходу в одному з автоматів. У результаті цього змінюється поточний стан даного автомата (I) і стан одного з каналів $K(I, J)$ або $K(J, I)$ в залежності від типу події, що виникає в автоматі I (прийом або передача). Стан каналів залишається незмінним, якщо виникає внутрішня (наприклад, Таймерна) подія.

По закінченні генерації нових глобальних станів створюється граф, вузли якого є частковими глобальними станами, а дуги – можливими переходами між цими станами. Аналіз такого графа надає можливість перевірити наступні властивості протоколу:

– втрату або неспецифікований прийом вхідної події (властивість 2). Канал $K(I, J)$ містить подію, а в автоматі I немає переходу з поточного стану, поміченого прийомом цієї події;
– тупиковий стан (властивість 1). Для всіх I $I = 1 \ldots N$ поточний стан $G(I)$ не має переходів, помічених вихідними або внутрішніми подіями, а всі канали пусті;

– переповнення середовища (властивість 5). Виникає в тому випадку, якщо серед всіх автоматів знайдеться хоча б один такий, в якому є перехід з поточного стану, помічений вихідною подією в канал $K(I,J)$, і число подій в цьому каналі дорівнює $Cap(I,J)$;
– надлишкова специфікація (властивість 4). Виникає в тому випадку, якщо в якому-небудь автоматі знайдеться перехід, котрий ні разу не був використаний при побудові графа досяжних глобальних станів.

В якості прикладу розглянемо побудову графа досяжних станів для двох автоматів (рис. 6.8). Функціонування кожного автомата представлено у вигляді графа, вузли якого є станами автомата, а направлені дуги – переходами між цими станами. Рис. 6.9 демонструє приклад простої процедури встановлення і роз'єднання зв'язку між двома протокольними автоматами. Взаємодія відбувається шляхом обміну неподільними елементами, прийом і передача яких є подіями для автоматів, що змінюють їх стан. Всі події перенумеровані цілими числами. Знак цілого використовується для вказівки типу події. Від'ємні цілі представляють події передачі, а позитивні – події прийому. Обидва автомата починають взаємодіяти в стані 0 і можуть видати вимогу ВСТАНОВИТИ_З'ЄДНАННЯ (-2), переходячи при цьому в стан 1. Роз'єднання з'єднання може почати тільки автомат B. Граф досяжних станів для розглянутого прикладу зображений на рис. 6.9. Символом λ на рисунку позначені стани "пустих" каналів. Отриманий граф досяжних станів демонструє помилки типу неспецифікованого прийому та переповнення каналу (передбачається, що $Cap(1,2)=1$.

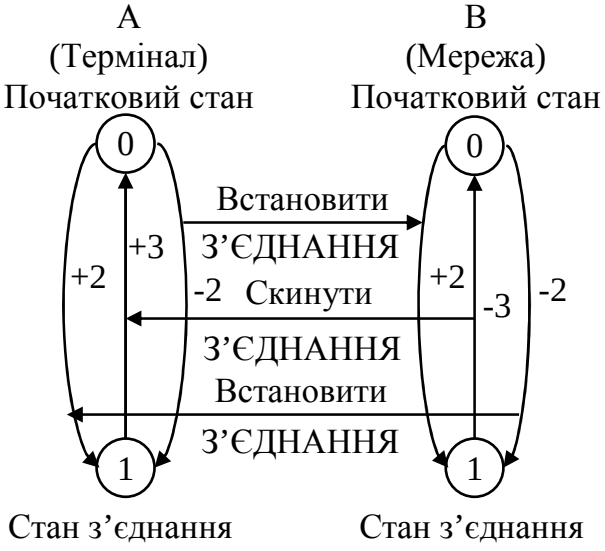


Рис. 6.8. Простий протокол встановлення з'єднання

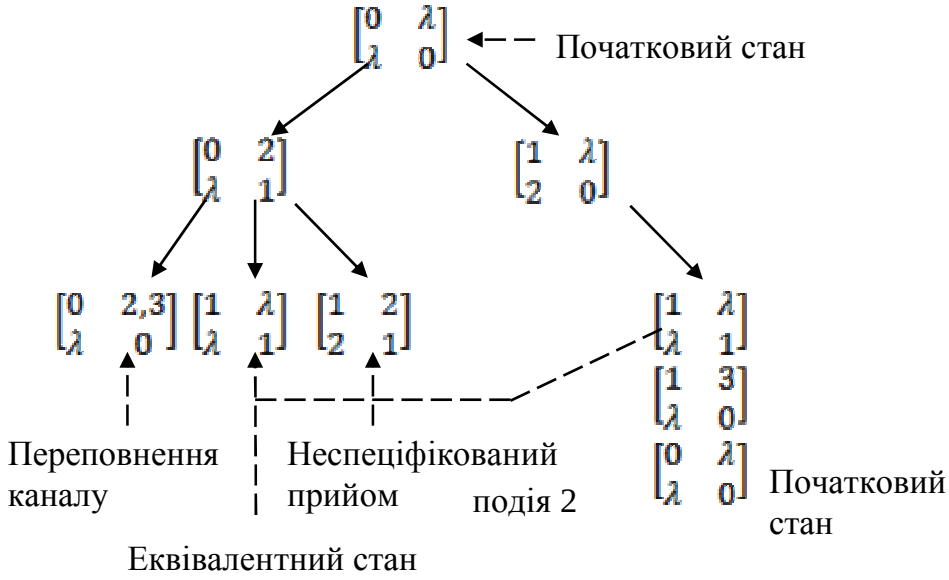


Рис. 6.9. Граф досяжних станів простого протоколу встановлення з'єднання

Метод діалогових матриць. В цьому методі послідовності взаємодій між протокольними автоматами представляються парами шляхів на графах і звуться *"діалогами"*. Шлях на кожному графі, званому *"монологом"*, повинен починатися і закінчуватися в початковому стані (стан 0) і не повинен містити проміжних переходів через цей стан. Монологи представлені цілими, якими помічені дуги, що входять до монологу. Для отримання всіх монологів для будь-якого графа був розроблений спеціальний алгоритм. Позначимо множину всіх монологів графа A через S_A, а графа B через S_B в. Позначимо потужність множини M через $|M|$.

Нехай $|S_A| = m$, а $|S_B| = n$. Тоді діалоги автоматів A і B можна представити у вигляді множини пар $[A_i, B_k]$, причому $A_i \in S_A$, $B_k \in S_B$, $i = 1 \ldots m$, $k = 1 \ldots n$. Легко помітити, що всі діалоги представляють декартовий добуток $S_A \times S_B$ *і можуть бути записані у вигляді матриці*

$$S_A \times S_B = \begin{bmatrix} [A_1B_1] & \ldots & [A_1B_k] & \ldots & [A_1B_n] \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ [A_iB_1] & \ldots & [A_iB_k] & \ldots & [A_iB_n] \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ [A_mB_1] & \ldots & [A_mB_k] & \ldots & [A_mB_n] \end{bmatrix} \qquad (6.6)$$

Були сформульовані критерії правильності діалогів. Діалог вважається правильним, якщо для нього *виконуються наступні умови:*

– кожна подія, що видається в одному монолозі, повинні прийматися в іншому (післяумова);
– кожна подія, прийняте в одному монолозі, має бути попередньо видана в іншому (передумова);
– якщо можлива одночасна передача подій $-X$ і $-Y$, видаваних у двох різних монологах, то взаємодія повинна проектуватися так, щоб подія X/Y була прийнята незалежно від того, видано чи ні подія Y/X (умова повноти).

На рис. 6.10 показаний скоригований протокол. Корекція звелася до додавання на графі A додаткового стану для вирішення конфлікту передачі події -2 на користь автомату A. При цьому діалогова матриця стала мати розмірність 3×3, а функція не містить помилкових взаємодій і має вигляд

$$Val\left[S'_A \times S'_B\right] = \begin{bmatrix} 1\,0\,0 \\ 0\,1\,0 \\ 0\,0\,1 \end{bmatrix}, \qquad (6.7)$$

де S'_A, S'_B – множина скоригованих монологів графів A і B відповідно.

Врахування помилок середовища передачі (дублювання, втрати, спотворення і т.д.) здійснюється в даному методі шляхом додавання відповідних подій і станів на графах автоматів.

Метод дозволяє перевірити властивості 1…4

Метод фазових діаграм також відображає діалоговий підхід до перевірки протоколів, але використовує графічну техніку для аналізу взаємодій. Метод дозволяє перевірити властивості 1 ... 5.

Метод "прилеглих станів". Деякі переходи в різних автоматах можна зв’язати один з одним і вважати, що вони виконуються тільки сумісно, тоді як решта не зв’язані один з одним переходи виконуються незалежно. Метод використаний для перевірки протоколу X.25 і дозволяє перевірити властивості 1 ... 8.

Метод сумісних шляхів. Даний метод розроблений для виявлення статичних блокувань. Він полягає в тому, що спочатку визначаються ситуації блокування, а потім робиться перевірка того, чи можуть вони бути реально досягнуті. Такий підхід дозволяє аналізувати не всю сукупність глобальних станів, а тільки ту підмножину, яка може привести до блокування.

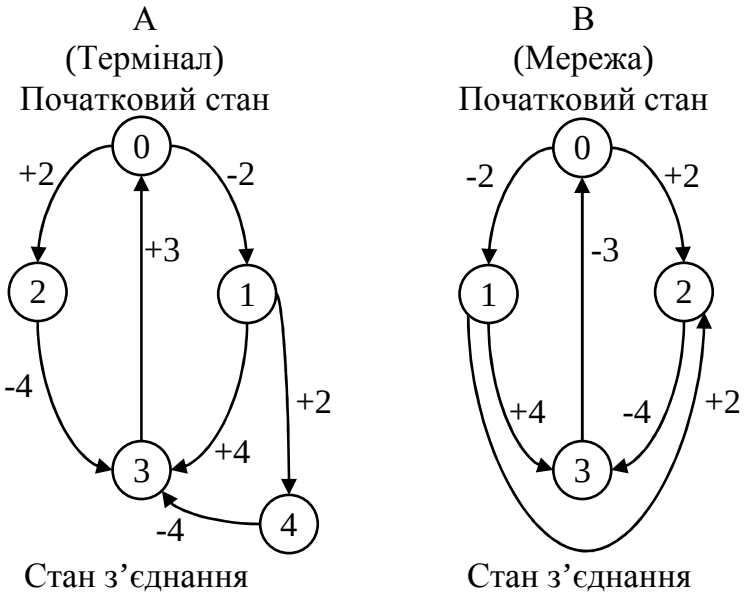


Рис. 6.10. Виправлений протокол встановлення з'єднання

Два шляхи *називаються сумісними*, якщо вони можуть бути розроблені одночасно протокольними автоматами, що взаємодіють через середовище. Ступінь спільності шляхів залежить від характеристик передавального середовища так, що погіршення характеристики збільшує ступінь сумісництва.

Аналіз специфікацій на основі формальних граматик. Техніка перевірки протоколу, описаного за допомогою формальної граматики, складається з двох етапів.

Перший етап:

- синтаксична перевірка специфікації, яка може бути виконана на етапі компіляції. Вона включає в себе перевірку наступних умов: у граматиці не повинно бути лівосторонньої рекурсії, а початкові символи альтернатив в породжуючих правилах повинні відрізнятися. Виконання цієї умови дозволяє скористатися стандартними методами спадного розбору без повернень;

- в граматиці не повинно бути породжуючих правил;

- при наявності в граматиці породжуючих правил, що визначають цикли, в граматику вводять обмеження на кількість повторів цих циклів з метою попередження нескінченних циклів (наприклад, обмежується кількість повторних передач).

Відсутність синтаксичних помилок у граматиці ще не гарантує коректності протокольних властивостей, тому пропонується другий етап перевірки, заснований на аналізі досяжних глобальних станів і використовуючий модель *"перевірочного автомата"* VA (Validation Automat). Ідея полягає в представленні каналу між взаємодіючими

автоматами у вигляді матриці черг повідомлень і підтверджень, а також визначень операцій роботи з чергами, котрі в свою чергу, можуть змінити стан системи. Перевірочний автомат VA отримує з граматики дій шляхом заміни всіх термінальних дій на відповідні операції.

Метод дозволяє зручно моделювати властивості передаючого середовища і перевірити властивості 1 ... 6.

Розвиток методів аналізу досяжних станів. Як уже зазначалося, істотним недоліком методу аналізу досяжності є швидке зростання числа глобальних станів при переході до складних протоколів. Останнім часом розроблено ряд методів, в котрих для розв'язання проблеми розмірності пропонуються наступні шляхи:

1 – декомпозиція, заснована на розбитті протокольної моделі на підмоделі (фази, функції або модулі);

2 – редукція, заснована на аналізі попередньо створеної обмеженої множини еквівалентних в деякому розумінні станів.

Розроблений метод, який дозволяє вважати ряд послідовностей взаємодій протокольних автоматів як еквівалентні, що забезпечується за рахунок введення певних правил впорядкування, а саме всі еквівалентні послідовності характеризують одні й ті ж кроки, що виконують автоматами при переході від одного стабільного стану до другого, і відрізняються тільки порядком виконання переходів у часі. Кожна підмножина еквівалентних послідовностей має свого представника – канонічну послідовність. Описаний алгоритм генерації канонічних послідовностей для двох взаємодіючих автоматів;

3 – обмеження числа виявлених помилок. Методи цього напрямку передбачають виявлення тільки помилок з обмеженої множини їх видів. Це пояснюється тим, що в результаті введення канонічних послідовностей не можуть бути виявлені помилки, котрі визначаються порядком виконання переходів взаємодіючих автоматів (динамічні блокування і переповнення каналів);

4 – побудова дерева досяжних станів для кожного із взаємодіючих автоматів замість загального дерева досяжних глобальних станів. При цьому розмірність дерев (графів) обмежується параметрами, що задає користувач. Оцінка складності обох підходів в гіршому випадку представляє собою експоненційну залежність, але метод дає меншу експоненту.

В основі метода лежить можливість шляхом аналізу логіки функціонування одного з взаємодіючих автоматів робити правильні висновки щодо результатів роботи всього протоколу. Завершеність алгоритму гарантується для випадку двох взаємодіючих автоматів і каналів кінцевої ємності.

6.6.2. Перевірка мереж Петрі

Звичайні мережі Петрі. Для звичайної мережі Петрі (МП) існує добре розроблений апарат аналізу її властивостей [10]. Встановивши відповідність між цими властивостями і властивостями коректності протоколу, можна отримати засоби аналізу протокольних специфікацій. Зробимо неформальний опис властивості і визначення МП.

Перехід t називається живим для розмітки M, якщо з будь-яких розміток M', досяжних з M, існує послідовність, що містить перехід t. Мережа Петрі вважається живою, якщо всі переходи живі.

Мережа Петрі називається *обмеженою*, якщо, для будь-якого $p \in P$ існує ціле $K_i > 0$, таке, що для всіх досяжних розміток виконується $M(p_i) \le K_i$. Якщо для всіх позицій $K_i = 1$, то мережа *називається безпечною*.

Послідовність спрацювання переходів σ називається *повторюваною*, якщо будь яка розмітка, що породжує σ, породжує також $\sigma' = \sigma\sigma\ldots$. Якщо множина переходів T', що входять до σ, дорівнює множині переходів мережі T, то МП називається повторюваною.

Можна показати, що якщо МП жива і обмежена, то властивості коректності відповідного протоколу будуть задовольнятися.

Існує три підходи до аналізу властивостей МП:

– побудова графа досяжних розміток;

– дослідження структури графа МП;

– доведення інваріантів.

Аналіз властивостей МП за графом досяжності дає найбільш повну характеристику, оскільки граф досяжності перераховує всі розмітки і всі послідовності спрацьовування переходів. Властивість живості розпізнається за наявністю для кожного $t \in T$ нескінченої послідовності спрацювань. Властивість обмеженості – по відсутності двох розміток M' і M'', що належать одній і тій же послідовності спрацювань переходів, таких, що $M' > M''$. *Крім того, даний метод дозволяє визначити, чи досяжна задана розмітка.*

Використання методу аналізу графа досяжних розміток можливо тільки в тому випадку, якщо граф кінцевий.

Методи, що використовують аналіз структури МП, відомі для *класу мереж з вільним вибором[14]*.

Використання методу доказу інваріантів для аналізу властивостей МП полягає в наступному. *Вводиться матриця інцидентності мережі* розмірності $m \times n$ (m – число позицій, n – число переходів), елементи котрої C_{ij} визначаються наступним чином:

$$C_{ij} = \begin{cases} v(p_i, t_j), \text{ якщо } p_i \in t' \\ -v(t_j, p_i), \text{ якщо } p_i \in t' \\ 0, \text{ якщо } p_i \text{ і } t_j \text{ непов'язані дугою}, \end{cases} \quad (6.8)$$

де $v(p_i, t_j)$ – кратність дуги (p_i, t_j); t (t') – множина вершин, інцидентних вхідним (вихідним) дугам. Визначаються T- і P-інваріанти, котрі перелічують послідовності переходів і підмножин позицій відповідно, зберігаючи зважену суму міток при всіх досяжних розмітках. Доказ інваріантів виконується за допомогою алгебраїчних операцій над матрицями інцидентності. Самі інваріанти виводяться з аналізу структури мережі і використовуються для перевірки загальних властивостей коректності протоколів.

Часові мережі Петрі. Аналіз загальних властивостей ЧМП виконується за допомогою графа досяжних розміток, як і для звичайної МП. Введення часових обмежень на спрацювання переходів суттєво збільшує виразну потужність мережі, але в загальному випадку робить *нерозв'язуваною проблему обмеженості і живості*.

Розширені мережі Петрі. Для перевірки властивостей МП-ЦМ використовується аналіз графа досяжних розміток, оскільки правила спрацювання МП-ЦМ дозволяють побудувати такий граф. Проблеми аналізу графа виникають лише у разі його великої розмірності, тому що, незважаючи на те, що МП-ЦМ дозволяє відобразити систему компактно, число станів системи може бути досить велике.

Перевірка властивостей МП-ПП і МП-ПД виконується за допомогою *методу доказів інваріантів* у чистому вигляді або в комбінації з аналізом графа досяжних розміток. Зокрема, управляюча структура може бути перевірена тими ж засобами, що і звичайна МП. При перевірці всієї системи і в цілому можна скористатися наступною особливістю. Додавання предикатів до МП може тільки зменшити число досяжних розміток. Таким чином, *обмеженість звичайної МП* тягне обмеженість предикатної МП. Це не стосується властивості живості, доказ якої для МП-ПП представляє значні труднощі.

6.6.3. Перевірка коректності регулярних виразів

В цьому методі регулярні вирази використовуються для опису поведінки кожного протокольного об'єкта, а їх взаємодію визначають

спеціальним "крос-оператором". Отримані в результаті використання крос-оператора вирази називають протокольними.

Даний метод дозволяє перевіряти властивості 1 ... 4.

6.6.4. Синтез протоколів

Альтернативним підходом до перевірки на коректність існуючої специфікації є *синтез завідомо коректних протоколів*. Під синтезом розуміють побудову специфікації протоколу за допомогою правил, що гарантують виконання деякої наперед заданої множини властивостей. *Методи синтеза протоколів* найчастіше основані на покроковому процесі доповнення діаграм станів кінцевого автомата, що використовуються для специфікацій протоколів.

Відомо декілька підходів до синтезу протоколів. В першому підході розроблені три правила проектування протоколу, що описує взаємодію двох автоматів через ідеальні канали FIFO (тобто в каналах відсутні помилки, пов'язані з втратою, спотворенням, дублюванням або розупорядкуванням). У синтезованому протоколі гарантується виконання властивостей 1 ... 5. Процес синтезу полягає в добудові спрямованих графів, що специфікують протокольні автомати, з використанням інтерактивної взаємодії з проектувальником. Втручання проектувальника потрібно кожного разу, коли в специфікації одного автомата необхідно визначити нову взаємодію. Додаткові стани і переходи в специфікації іншого автомата, які є відповідною реакцією на нову взаємодію, добудовуються автоматично відповідно до правил проектування. В алгоритмі синтезу сформульовані умови, при яких процес побудови графів завершується. В *запропоновано розширення методу для більшої кількості взаємодіючих автоматів*.

В іншому методі розроблені правила, що дозволяють розглядати неідеальні канали, що відрізняються від FIFO. В процесі синтезу одночасно будуються діаграми станів для кожного об'єкта і дерево глобальних станів для протоколу в цілому. *Глобальні стани описані квадратною матрицею*, діагональні елементи котрої відповідають станам автоматів, а решта – станам черг, моделюючих канали між автоматами. Пропонуються чотири правила синтезу протоколів. *Два перших* визначають алгоритм, відповідно якому при прийомі і передачі повідомлень в діаграму станів протокольного автомата додаються нові вузли. *Третє правило* описує вплив типу каналу на конструювання згаданої діаграми станів (враховується послідовність

надходження повідомлень з каналів). Відповідно четвертому правилу після використання кожного з перших трьох правил дерево глобальних станів необхідно перевірити відносно заданої множини властивостей. В результаті забезпечується виконання в синтезованому протоколі властивості 1, 2, 4 ... 7. Обидва розглянутих вище методи в якості початкової інформації використовують неформальний опис протоколів.

В третьому підході [1] на основі відомої специфікації в цілому сформульовані правила побудови специфікації нового модулю, якого не вистачає. Наприклад, може бути задана специфікація, наданого протоколом сервісу і специфікація одного з протокольних об'єктів. Метод дозволяє отримати специфікацію іншого протокольного об'єкта так, щоб специфікації обох протокольних об'єктів в цілому були узгоджені з специфікацією сервісу, тим самим забезпечується виконання властивостей протоколів 1, 2, 4, 5, 7.

6.7. Методи верифікації

Як вже зазначалось, *під верифікацією розуміється* доказ того, що специфікація протокольних об'єктів рівня, що досліджується разом з специфікацією сервісу нижніх рівнів, які використовується даними об'єктами, узгоджена з описом сервісу, що надається цим рівнем протоколу Задача верифікації більш складна і менш дослідження, ніж задача аналізу коректності. *Нижче наведені окремі методи верифікації протокольних систем*. Спочатку розглянемо можливість застосування для верифікації методу аналізу досяжних станів, а потім – методів логічного доказу.

6.7.1. Використання аналізу досяжних станів для верифікації протоколів

Метод аналізу досяжних станів забезпечує достатньо потужний і *універсальний механізм для аналізу протокольних систем*. Раніше була описана методика його використання для перевірки коректності специфікацій.

Розглянемо протокольні специфікації об'єктів рівнів $(N-1)$, N і $(N+1)$. Процес верифікації складається з трьох кроків.

Перший крок включає перевірку обмежень, що накладаються використовуваним сервісом. Для його виконання необхідно: виділити проекцію специфікації N-об'єкта на $(N-1)$ використаний сервіс, виділити проекцію специфікації $(N-1)$ -об'єкта на N- наданий сервіс, отримані проекції перевірити методом аналізу досяжних станів на коректність взаємодії.

Другий крок аналогічний перевірці локальних обмежень, накладених наданим сервісом. Він складається з того, щоб виділити проекцію специфікації N-об'єкту на N- наданий сервіс, виділити проекцію специфікації $(N+1)$ - об'єкти на N- використаний сервіс і виконати верифікацію отриманих проекцій методом аналізу досяжних станів.

Третій крок полягає в аналізі сукупності специфікацій локального N і віддаленого N' об'єктів з урахуванням сервісних примітивів. Для цього будується дерево досяжних станів, з котрого крім перевірки властивостей коректності можна отримати множину глобальних сервісних послідовностей. Далі ця множина порівнюється з допустимою множиною сервісних послідовностей і робиться висновок щодо узгодженості специфікацій N-об'єктів використаного і наданого сервісу між собою. Допустима множина сервісних послідовностей визначається або методом проекції дерева глобальних станів об'єктів-користувачів на використаний сервіс, або стандартом на сервіс.

6.7.2. Метод логічної індукції по числу подій

Метод оснований на використанні математичної індукції і отримав розвиток при доказі правильності програм[15]. *Метод оснований на* деяких аксіомах або правилах верифікації. Для запису правил верифікації використовується наступна формула:

$$\{A_i\} S \{A_j\}, \tag{6.9}$$

де A_i і A_j – індуктивні твердження, а S – фрагмент специфікації протоколу. *Такий запис значить, що якщо безпосередньо перед* виконанням S справедливе A_i (передумова), то після виконання S (якщо воно закінчено) справедливе A_j (післяумова).

Якщо у вигляді тверджень виразити вимоги, пред'явлені до сервісу, наданому протоколом, то, довівши ці твердження для всіх можливих дій і переходів протокольної системи, ми виконаємо її верифікацію.

Часто твердження, що визначають вимоги сервісу, повинні бути справедливими незалежно від подій і переходів протокольного объекту. Такі твердження називаються *інваріантними системами*. Для доказу інваріанта I використовується метод індукції за числом подій, який полягає в наступному [15]:

– перевіряється виконання I_0, тобто справедливість твердження I для початкового стану системи;
– для переходу t, що переводить систему зі стану P в стан P', перевіряється, що $I_p\ P_t\{S_t\} I'_p = істина$, де P_t – предикат, що дозволяє перехід t, а S_t – послідовність дій, пов'язаних з виконанням переходу t. Іншими словами, перевіряється, що якщо твердження істинне для деякого стану P системи і виконується перехід t, котрий переводить систему в стан P' і виконується при цьому дія S_t, то істинність твердження зберігається.

Дана задача аналогічна традиційній задачі, що розв'язується при доказі правильності програм методом індуктивних тверджень, якщо перехід t порівняти з деякою точкою програмної специфікації, а послідовність дії S_t – з послідовністю програмних операторів.

Важливо відзначити, що даний метод дозволяє здійснити верифікацію тільки часткової коректності. Тобто при роботі протоколу гарантується його коректне функціонування, проте не розглядається така важлива проблема, як досяжність протоколом певних станів.

6.7.3. Використання часової логіки

Іншим способом верифікації повної коректності є використання в інваріантах і твердженнях виразів часової логіки. Так, повна коректність може бути доведена, якщо показати, що специфікація має властивості безпеки і живості.

6.7.4. Комбінаторні методи

Як вже відмічалось, метод логічної індукції дозволяє виконати лише часткову верифікацію. Для повноти верифікації метод можна об'єднати з аналізом досяжних станів. При цьому розвиток системи відображається в досяжних його станах, а вимоги до системи (сервіс) у твердженнях. Такі комбіновані методи зручно застосовувати для верифікації специфікацій, заснованих на моделях розширених автоматів, тому що вони мають як властивості автомата, так і звичайних програм.

6.7.5. Модель управління взаємодією відкритих систем

Виділяють три категорії управління ресурсами ВВС:

1) управління системами;
2) управління N-рівнем;
3) операції N-рівня.

Управління системами забезпечує механізми контролю, управляючих дій і координації всіх ресурсів ВВС всередині операційної системи. Воно забезпечує управління ресурсами ВВС, які стосуються до одного або декількох рівнів. Управління системами є єдиним засобом, що забезпечує управління сукупністю рівнів. Передача даних при управління системами здійснюється за допомогою протоколів прикладного рівня. *Функції управління системами*, що локалізовані у відкритій системі, реалізуються прикладним процесом управління системами, частина якого, що стосується передачі даних, називається прикладним об'єктом управління системами.

Управління N-рівнем забезпечує механізм контролю, управляючих дій і координації ресурсів ВВС N-рівня. Управління N-рівнем може діяти на сукупність елементів зв'язку. Об'єкти управління N-рівнем взаємодіють один з одним для забезпечення управляючих дій на ресурси ВВС N-рівня, що використовуються для передачі даних між відповідними відкритими системами. Для передачі даних з метою управління N-рівнем використовуються протоколи управління системами, протоколи управління N-рівнем або протоколи обох категорій.

Операції N-рівня забезпечують сукупність засобів управляючих дій і управління одним елементом зв’язку. Тільки один елемент зв’язку всередині одного рівня може підлягати дії цього типу управління ВВС.

6.8. Тестування протокольних реалізацій

Незважаючи на розвиток методів формальних описів, аналізу коректності та верифікації протоколів і сервісів, тестування грає важливу роль в погоджені розроблених в рамках мережевих архітектур різнорідних пристроїв і програмно-апаратних реалізацій протоколів. Це пояснюється тим, що багато реалізацій поки що отримуються "ручними" способами. Крім того, навіть реалізації, отримані автоматично з специфікацій, функціонують в різнорідних програмно-апаратних

середовищах і операційних системах, що залишаються за межами області специфікації.

Архітектура і способи застосування систем тестування в розподіленому оточенні відрізняються від традиційних систем тестування. Щоб усунути розбіжності в методах тестування різних дослідницьких груп і виробити загальну методологію, в ISO була створена цільова робоча група з тестування реалізацій, що виконуються в рамках еталонної моделі взаємодії відкритих систем (модель OSI).

Спочатку до засобів тестування були розроблені загальні вимоги. Ці засоби повинні бути основані на використанні стандартних специфікацій, бути достатньо загальними, *щоб бути незалежними від окремих реалізацій*, модульними, щоб адаптуватися до вимог користувачів, і простими у використанні.

При виборі архітектури тестування звичайно враховують два основних фактори:

1) реалізація, що тестується (ТР) повинна розглядатися як "чорна скриня", тобто без урахування її внутрішньої структури,

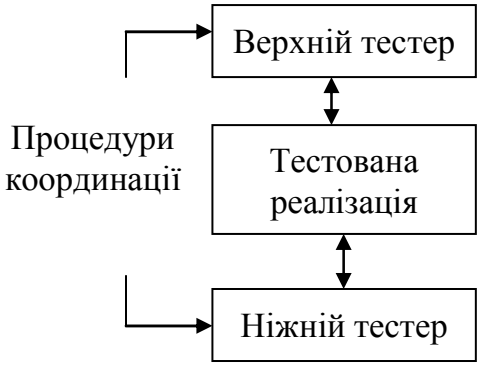
2) система тестування повинна бути розподіленої і забезпечувати можливість тестування на віддаленій ЕОМ.

Наслідком цих факторів явилося те, що розробляються архітектури тестування, які також, засновані на еталонній моделі OSI.

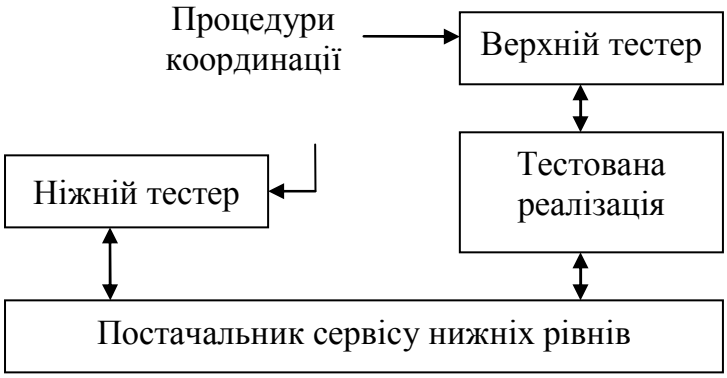
Розроблена абстрактна методологія тестування, яка розглядає три групи – локальні, розподілені і віддалені методи тестування. Кожна група може розглядати ТР як об'єкт одного рівня або сукупність об'єктів декількох суміжних рівнів (рис. 6.11).

Після визначення абстрактної методології тестування проводять інтенсивні роботи в області розробки методів генерації послідовностей тестових подій, в якості яких використані неподільні елементи тестування на обраному рівні абстракції.

Тестування системи в широкому сенсі складається з перевірки того, що її поведінка на концептуальній межі між ТР і оточенням відповідає передбачуваному. Розрізняють вхідні тестові події (від оточення до ТР) і вихідні тестові події (від ТР до оточення). В принципі вхідні події можуть подаватися оточенням недетерміновано, але в дійсності поведінка систем, що проектується передбачає певну поведінку оточення, тому конкретна тестова подія (вхідна або вихідна) визначає можливі наступні події. Як наслідок, виникає необхідність опису абстрактної поведінки оточення і ТР для отримання тестових сценаріїв в термінах абстрактних примітивів сервісу і елементів протокольних даних.



а)



б)

в)

Рис. 6.11. Локальний (а), розподілений (б) и віддалений (в) методи тестування

Тестовий сценарій можна розглядати як об'єднання окремих тестів, кожний з яких містить множину подій для досягнення визначеної мети (наприклад, встановлення з'єднання, відміни з'єднання і т.д.). Об'єднання вхідних і викликаних ними вихідних подій розглядається в тесті як неподільне ціле. Якщо специфікація допускає впорядкування вхідних подій у часі, то тест вважається послідовним.

6.9. Адміністративне управління інформаційно - обчислювальними мережами

Як і більшість стандартів міжнародної організації зі стандартизації (ISO), стандарти управління взаємодією відкритих систем (OSI) регламентують процедури, що безпосередньо стосуються передачі інформації. В рамках таких процедур в управлінні OSI визначені основні інформаційні служби, що забезпечують обмін інформацією в інтересах п'яти виділених груп функцій управління, названих дисциплінами управління. Об’єднання функцій управління серед виділених дисциплін обумовлене тими аспектами функціонування відкритих систем, з якими пов'язані дані функції управління.

6.10. Архітектура управління взаємодією відкритих систем

6.10.1. Концепції управління OSI

Архітектура управління взаємодією відкритих систем визначає ВВС: термінологію і основні принципи концепції управління; модель управління і склад служб управління; особливості управління ВВС.

Функціональне оточення управління ВВС являє собою підмножину загального функціонального оточення ВВС, що відноситься до засобів і служб управління взаємодією і ресурсами ВВС. Функціональне оточення управління ВВС забезпечує засоби як для збирання даних і здійснення управляючим персоналом, так і для обробки повідомлень щодо подій і звітів про стан ресурсів ВВС.

До ресурсів ВВС належать: засоби, необхідні для функціонування протоколів ВВС, і дані управління, що забезпечують інформацією про стан функціонального оточення ВВС.

В рамках управління ВВС виділяються наступні дисципліни:

1) управління при відмовах;
2) управління обліком;
3) управління конфігурацією і іменами;
4) управління ефективністю функціонування;
5) управління безпекою.

Управління при відмовах – сукупність засобів, ініційованих у результаті ненормальної роботи функціонального оточення ВВС.

Відмови проявляються у вигляді збоїв при функціонуванні операційної системи. Управління відмовами надає засоби для обслуговування і аналізу файлів реєстрації збоїв, прийому і обробки повідомлень щодо виявлення збоїв, адміністративного супроводження збоїв, виконання послідовності тестів, виправлення відмов.

Управління обліком – сукупність засобів, що забезпечують визначення вартості ресурсів і сплати за їх використання. Управління обліком надає засоби для оповіщення користувача про сплату або об'єм використаних ресурсів, встановлення облікових лімітів на використання ресурсів, визначення вартості використання сукупності ресурсів.

Управління конфігурацією та іменами – сукупність засобів управління, ідентифікації, збору і надання даних, що забезпечують безперервне функціонування служб взаємодії. Включає засоби установки параметрів відкритих систем, ініціалізації і закриття ресурсів ВВС, збору даних про стан відкритих систем, забезпечення конкретними даними за запитом.

Управління ефективністю функціонування – сукупність засобів, необхідних для оцінки поведінки ресурсів ВВС і ефективності діяльності щодо взаємодії. Сюди відносять збір статистичних даних, необхідних для обслуговування і аналізу файлів реєстрації станів систем.

Управління безпекою – сукупність засобів захисту ресурсів ВВС, тобто засобів санкціонування, контролю доступу, шифрування і управління ключами, аутентифікація, обслуговування та аналізу реєстраційних файлів безпеки.

6.10.2. Обмін інформацією управління

Розрізняють три категорії обмінів інформацією управління ВВС:

1) управляючі дії;
2) передача інформації;
3) повідомлення про події.

Обмін інформацією управління розглядається як двостороння взаємодія, в якій сторони виконують ролі ініціатора або відповідача для кожного одиничного акту обміну.

Ініціатор формує управляючі дії, тобто видає запит на певну функцію управління. Відповідач формує відповідь на управляючий запит.

Діяльність щодо передачі інформації реалізується в запитах, що видає ініціатор, на певну інформацію і в відповідях, що формує відповідач.

Повідомлення щодо подій формуються і посилаються ініціатором. Якщо вимагається підтвердження на прийом такого повідомлення, то воно формується відповідачем у вигляді відповіді.

6.10.3. Функціональна повнота управління

При умові функціональної повноти на всіх семи рівнях відкритої системи OSI (прикладний, представницький, сеансовий, транспортний, мережевий, канальний, фізичний) для взаємозв'язку прикладних об'єктів управління системами повинні використовуватися засоби управління системами.

При умові функціональної повноти на рівнях від 1 до $(N-1)$ для взаємозв'язку між об'єктами управління N-рівнем повинні використовуватися засоби управління цим рівнем.

Для повноти використання засобів управління системами відкрита система повинна мати відповідну функціональну повноту на всіх семи рівнях. Інакше функціональна повнота управління даною відкритою системою обмежена сукупністю окремих засобів управління рівнями даної відкритої системи.

Для підтримки управління N-рівнем відкрита система повинна мати функціональну повноту на рівнях від 1 до $(N-1)$. Якщо такої повноти немає, і відсутня підтримка для прикладних об'єктів управління системами, то функціональна повнота управління, що забезпечується даною відкритою системою, обмежена засобами управління, що забезпечують протоколи $(N-1)$ рівня даної відкритої системи.

Прикладні об'єкти управління системами можуть бути присутні у відкритій системі незалежно від існування об'єктів управління N-рівнем на кожному з рівнів.

6.10.4. Інформаційна база даних управління (ІБДУ)

ІБДУ являє собою множину даних управління ВВС у відкритій системі, доступних для функціонального оточення ВВС. Передача цих даних між відкритими системами здійснюється з використанням протоколів управління ВВС.

6.10.5. Модель потоку керуючих впливів

Процеси, що забезпечують управління ВВС, отримують управляючі дії:

1) від людей і (або) програмного забезпечення, що функціонують в якості адміністративних агентів, локальних для цього процесу управління;

2) від віддалених систем через їх прикладні об'єкти управління системами, об'єкти управління N-рівнем, об'єкти N-уровня.

Абстрактні синтаксис і семантика потоку управляючих дій всередині функціонального оточення ВВС визначаються синтаксисом і семантикою протоколів ВВС.

6.10.6. Модель потоку даних і ІБДУ

Вся інформація управління ВВС всередині відкритої системи є частиною ІБДУ цієї системи. Інформація формується і використовується:

1) локальними агентами управління;

2) віддаленими відкритими системами через протоколи: управління, управління N-рівнем, N-рівня.

6.11. Архітектура забезпечення безпеки

Архітектура забезпечення безпеки розширює область застосування еталонної моделі ВВС при передачі даних між відкритими системами [1]. Вона містить загальний опис служб безпеки і відповідних механізмів, що забезпечуються еталонною моделлю, визначає розташування даних служб і механізмів у ВВС.

6.11.1. Служби безпеки

Служба аутентифікації однорівневих об'єктів забезпечує підтвердження справжності одного або декількох взаємодіючих об'єктів при встановленні з'єднання або періодично протягом фази передачі даних.

Служба аутентифікації джерела даних N-рівня забезпечує підтвердження $(N+1)$-рівню того, що відправником даних є об'явлений об'єкт $(N+1)$-рівня.

Контроль доступу забезпечує захист від несанкціонованого використання ресурсів. Цими ресурсами можуть бути як ресурси ВВС, так і інші ресурси, доступні через протоколи ВВС.

Служби секретності даних забезпечують їх захист від несанкціонованого розкриття.

Служба засекречування з'єднання забезпечує секретність всіх даних N-користувача при передачі по N-з'єднанню.

Служба засекречування без з'єднання забезпечує секретність всіх даних N-користувача, що передаються в одному блоці даних служби N-рівня без встановлення з'єднання.

Служба засекречування вибіркових полів забезпечує секретність вибіркових полів в даних N-користувача при передачі по N-з'єднанню або в одному блоці даних служби N-рівня без встановлення з'єднання.

Служба засекречування потоку даних забезпечує секретність інформації, що отримується при аналізі потоків даних (наявність, відсутність, об'єми, напрямок і інтенсивність потоків).

Служби цілісності з'єднання з відновленням направлена на забезпечення цілісності всіх даних N-користувача при передачі по N-з'єднанню з виявленням будь-якої модифікації, вставок, знищення і повторів будь-яких даних по всій послідовності блоків даних служби із можливістю відновлення.

Служба цілісності з'єднання без відновлення аналогічна службі цілісності з'єднання з відновленням, але не містить процедур відновлення.

Служба цілісності вибіркових полів з'єднання забезпечує цілісність вибіркових полів даних N-користувача в блоці даних N-служби при передачі по N-з'єднанню і приймає форму визначення факту модифікації, вставок, знищення і повторів вибіркових полів.

Служба цілісності без з'єднання забезпечує цілісність одного блоку даних N-служби без встановлення з'єднання і приймає форму визначення факту модифікації прийнятого блоку даних служби. Додатково можуть забезпечуватися обмежені форми виявлення вставок і повторів.

Служба цілісності вибіркових полів без з'єднання забезпечує цілісність вибіркових полів в одному блоці даних служби, що передаєть-

ся без з’єднання, і приймає форму визначення факту модифікації вибіркових полів.

Служба захисту від відмов може приймати одну або обидві наступні форми.

Служба захисту від відмов з підтвердженням джерела забезпечує одержувача необхідними доказами походження даних, що захищають від спроб відправника помилково відмовитися від факту посилки даних або від їх вмісту.

Служба захисту від відмов з підтвердженням доставки забезпечує відправника необхідними доказами доставки даних, що захищають від спроб одержувача помилково відмовитися від факту прийому даних або від їх вмісту.

6.11.2. Спеціальні механізми забезпечення безпеки

Наступні механізми можуть бути впроваджені в N-рівень для забезпечення послуг, розглянутих в п. 6.7.2.

Механізми шифрування. Шифрування може забезпечити секретність переданих даних і (або) інформацію щодо потоків даних. Виділяють два класи алгоритмів шифрування:

1) симетричне (з використанням секретного ключа шифрування і дешифрування);

2) асиметричне (з використанням ключа загального користування), при якому значення ключа шифрування не передбачає знання ключа дешифрування і навпаки. Два ключа таких систем зазвичай називають “ключем загального користування ” і “приватним ключем”.

Наявність механізму шифрування передбачає використання механізмів управління розподілом ключів.

Механізми цифрового (електронного) підпису, які включають процедури закриття блоків даних і перевірки закритого блоку даних.

Перший процес використовує приватну для закриваючої сторони інформацію (унікальну і секретну), другий – відкриті процедури і інформацію, які не дозволяють вивести приватну інформацію закриваючої сторони.

Процес закриття використовує шифрування блоку даних або формування криптографічної контрольної суми даних, що здійснюється за допомогою такої інформації, як приватний ключ.

Процес перевірки використовує відкриті процедури і інформацію для визначення відповідності підпису і приватної інформації закриваючої сторони.

Механізми контролю доступу здійснюють перевірку повноважень об'єкта на доступ до ресурсів. Використання механізмів контролю доступу ґрунтується на наступних елементах:

1) інформаційна база контролю доступу, в якій підтримується інформація щодо прав доступу до ресурсів повноважних об'єктів одного рівня;

2) інформаційна аутентифікація, така як паролі, розташування і форма представлення котрих є свідоцтвом повноважень здійснюючих доступ об'єктів;

3) санкції, розташування і форма представлення яких, є свідоцтвом права на доступ до об'єкту або ресурсу, заданому санкцією;

4) мітки безпеки, які, будучи пов'язаними з об'єктом, можуть бути використані для дозволу або відмові на доступ у відповідності до політики забезпечення безпеки;

5) час спроби доступу і (або) тривалість доступу;

6) тривалість і (або) маршрут спроби доступу.

Механізми контролю доступу можуть застосовуватися до будь-якого з учасників асоціації зв'язку і (або) в будь-якому проміжному об'єкті.

Механізми забезпечення цілісності даних. Є два аспекти цілісності даних: цілісність одного блоку даних або поля і цілісність потоку блоків даних або полів. В загальному випадку використовуються механізми, що забезпечують обидва типи цілісності даних.

Визначення цілісності одного блоку даних включає два процеси: на передаючому об'єкті і на приймаючому об'єкті. Передаючий об'єкт додає до блоку даних ознаку, значення якої є функцією від самих даних. Приймаючий об'єкт формує відповідне число і порівнює його з отриманою ознакою. Даний механізм не захищає від дублювання блоків даних.

Виявлення змін на відповідних рівнях архітектури забезпечення безпеки може викликати дії з відновлення на даному рівні або на рівні, що лежить вище.

Для передачі даних по з'єднанню захист цілісності послідовності блоків даних (усунення невпорядкованості, втрати, повторів і вставок або модифікації даних) вимагає додатково деякої форми послідовної

нумерації; в іншому випадку можуть використовуватися мітки часу або криптографічні ланцюги.

Для передачі в режимі без встановлення з'єднання обмежену форму захисту цілісності послідовності блоків даних можуть забезпечити мітки часу.

Механізми забезпечення аутентифікації. Для забезпечення аутентифікації використовуються паролі, що формуються передаючим об'єктом і перевіряються приймаючим об'єктом; криптографічні методи; перевірка характеристик об'єкта.

Механізми забезпечення аутентифікації можуть бути впроваджені в N-уровень з метою забезпечення аутентифікації об'єктів одного рівня.

Використані методи аутентифікації можуть поєднуватися з процедурою "триразового рукостискання ".

Вибір методу забезпечення аутентифікації залежить від умов, в яких він буде застосовуватися. В багатьох випадках ці механізми необхідно застосовувати разом з мітками часу і синхронізацією годинників; дво- і триразовим рукостисканням; службою захисту від відмов, що забезпечена механізмами електронного підпису і (або) огляду.

Механізми заповнення тексту можуть застосовуватися для забезпечення різного рівня захисту від аналізу потоків.

Механізми управління маршрутом. Маршрути можуть обиратися динамічно або бути заздалегідь заданими з тим, щоб використовувати тільки фізично безпечні субмережі, комутатори (ретранслятори мережевого рівня) або канали.

Кінцеві системи при виявлені постійних спроб нав'язування можуть вимагати від постачальника мережевих послуг встановлення з'єднання по іншому маршруту.

Дані, що містять певні мітки, можуть передаватися у відповідності з політикою забезпечення безпеки через задані субмережі, комутатори або канали. Крім того, ініціатор з'єднання (або відправник блоків даних, що передаються в режимі без встановлення з'єднання) може задавати маршрут в обхід конкретних субмереж, комутаторів або каналів.

Механізми огляду. Характеристики даних, що передаються між двома або більше об'єктами, такі як цілісність, джерело, час і одержувач, можуть підтверджуватися за допомогою механізму огляду. Дане підтвердження забезпечується третьою стороною, котрій довіряють обидві сторони і котра має необхідну інформацію.

6.11.3. Загальні механізми забезпечення безпеки

Загальними називають механізми забезпечення безпеки, які не стосуються певної служби N -рівня.

Гарантована функціональність. Гарантована функціональність повинна забезпечуватися для розширення можливостей або підвищення ефективності інших механізмів забезпечення безпеки. Будь-які функції, що реалізують механізми забезпечення безпеки, повинні заслуговувати довіру.

Виявлення і обробка подій. Різні механізми забезпечення безпеки можуть виявляти порушення умов забезпечення безпеки. Дії, що вживаються в такій ситуації, можуть включати: процедури відновлення; реєстрацію подій; місцевий звіт щодо події; віддалений звіт щодо події; одностороннє роз'єднання.

Запис звіту щодо перевірки безпеки. Перевірка безпеки представляє собою незалежну перевірку системних записів і діяльності на відповідність заданим режимам безпеки. В рамках ВВС регламентується тільки інформація, що реєструється, і специфікація форматів для обміну реєстраційними записами.

6.11.4. Ілюстрація взаємозв'язку служб і механізмів забезпечення безпеки

У табл. 6.3 представлений взаємозв'язок служб і механізмів забезпечення безпеки (знак "+" означає, що механізм може використовуватися самостійно або в комбінації з іншими механізмами).

6.11.5. Розміщення служб і механізмів безпеки

Розподіл служб і механізмів безпеки за рівнями інформаційного обміну OSI наведено в табл. 6.4.

Рівні інформаційного обміну OSI:

1 - фізичний;
2 - канальний;
3 - мережевий;
4 - транспортний;
5 - сеансовий;
6 - представницький;
7 - прикладний.

Таблиця 6.3

Взаємозв'язок служб і механізмів забезпечення безпеки

Найменування служби	Механізми (див. п. 6.2.2)							
	1	2	3	4	5	6	7	8
Аутентифікація однорівневих об'єктів	+	+			+			
Аутентифікація джерела даних	+	+						
Контроль доступу			+					
Засекречування з'єднання	+						+	
Засекречування без з'єднання	+						+	
Засекречування вибіркових полів	+							
Засекречування потоку даних	+					+	+	
Цілісність з'єднання з відновленням	+			+				
Цілісність з'єднання без відновлення	+			+				
Цілісність вибіркових полів з'єднання	+			+				
Цілісність без з'єднання	+	+		+				
Цілісність вибіркових полів без з'єднання	+	+		+				
Захист відмов з підтвердженням джерела		+		+				
Захист від відмов з підтвердженням доставки		+		+				

Таблиця 6.4

Розподіл служб і механізмів безпеки за рівнями інформаційного обміну OSI

Найменування служби	Рівні						
	1	2	3	4	5	6	7
Аутентифікація однорівневих об'єктів			+	+			+
Аутентифікація джерела даних			+	+			+
Контроль доступу			+	+			+
Засекречування з'єднання	+	+	+	+			+
Засекречування без з'єднання		+	+	+			+
Засекречування вибіркових полів							+
Засекречування потоку даних	+		+				+
Цілісність з'єднання з відновленням				+			+
Цілісність з'єднання без відновлення			+	+			+
Цілісність вибіркових полів з'єднання							+
Цілісність без з'єднання			+	+			+
Цілісність вибіркових полів без з'єднання							+
Захист відмов з підтвердженням джерела							+
Захист від відмов з підтвердженням доставки							+

6.12. Підсумковий коментар

Основою для створення сумісних дорогих програмно-технічних виробів, що можуть служити для розвитку складних розподілених систем, є застосування методів формального опису (МФО) протоколів і сервісів, які повно і однозначно визначають всі аспекти взаємодії.

Можна виділити два типи формальних специфікацій: специфікацію сервісів, які представляються протокольним рівнем, і специфікацію поведінки протокольних об'єктів в процесі надання сервісів. Незважаючи на різницю цих двох типів специфікацій, до методів їх опису в ISO були розроблені єдині вимоги. МФО повинні забезпечувати наступне: несуперечливу, ясну і точну специфікацію; основу для визначення повноти протоколів і сервісів, що описані; основу для аналізу протоколів і сервісів у відношенні коректності, ефективності ін.; засоби для верифікації того, що протоколи і сервіси відповідають вимогам архітектури, що розробляється; базу для визначення узгодженості стандартів один з одним; засоби для визначення відповідності реалізацій протоколів стандартам.

Розглянуто дві групи автоматних моделей і моделей послідовностей, що використовують МФО для специфікації протоколів і сервісів, а також деякі формальні методи аналізу і верифікації протоколів.

В технології розробки протоколів існує набір властивостей, обов'язкових для будь-яких протоколів. Тому перевірка коректності протоколу повинна забезпечувати доказ того, що специфікація протоколу має ці властивості. Властивості ці наступні: відсутність статичного блокування; повнота; однозначність відповідності станів; відсутність надлишковості; обмеженість; відсутність динамічного блокування; завершеність (розвиток); самосинхронізація. Розроблені різні класифікації протокольних властивостей.

Розглянуті особливості тестування розроблених в рамках мережевих архітектур різнорідних пристроїв і програмно-апаратних реалізацій протоколів. В ISO розроблена загальна абстрактна методологія з тестування реалізацій, що виконуються в рамках еталонної моделі взаємодії відкритих систем, яка розглядає три групи – локальні, розподілені і віддалені методи тестування.

Адміністративне управління інформаційно-обчислювальними мережами зумовлене стандартами (ISO) на управління взаємодією відкритих систем (ВВС) і регламентує процедури, що безпосередньо стосуються передачі інформації. В рамках таких процедур в управ-

лінні ВВС визначені основні інформаційні служби, що забезпечують обмін інформацією в інтересах п’яти виділених груп функцій управління.

Архітектура управління ВВС визначає: термінологію і основні принципи управління; модель управління і склад служб управління; особливості управління ВВС.

В рамках управління ВВС виділяються наступні дисципліни: управління при відмовах; управління обліком; управління конфігурацією і іменами; управління ефективністю функціонування; управління безпекою.

Розрізнюють три категорії обміну інформацією управління ВВС: управляючі дії; передача інформації; повідомлення щодо подій.

Виділяють три категорії управління ресурсами ВВС: управління системами; управління N-рівнем; операції N-рівня.

Архітектура забезпечення безпеки розширює область застосування еталонної моделі ВВС при передачі даних між відкритими системами. Вона містить загальний опис служб безпеки (аутентифікації, контроль доступу, секретності, цілісності, захисту від відмов) і відповідних механізмів (шифрування, цифрового (електронного) підпису, контролю доступу, забезпечення цілісності даних, забезпечення аутентифікації, заповнення тексту, управління маршрутом, огляду), що забезпечуються еталонною моделлю, визначає розташування даних служб і механізмів в ВВС.

6.12. Література до розділу 6

1. *Бохман Г.В., Мерлин Ф.М.* Разработка связных протоколов// Проблемы МСН-ТИ/МЦНТИ. – М., 1981. – №12, – С.146-155.
2. *Мизин И.А., Кулешов А.П.* Сети ЭВМ // Итоги науки и техники. Сер. Техническая кибернетика. – Т. 20. – М.: ВИНИТИ, 1986. – С. 3135.
3. *Пітерсон Дж.* Теорія мереж Петрі і моделювання систем. – М.: Світ, 1984. – 150 с.
4. *Jensen K.* Coloured Petri Nets and the Invariant Methods // Theoretical Computer Science – 1981. – Vol. 14, N 3. – P. 317-336.
5. Протоколы и методы управления в сетях передачи данных: Пер. с англ./ Под ред.Ф.Ф. Куо. – М.: Радио и связь, 1980. – 480с.

6. *Genrich H.J., Lautenbach K.* Thiagarajan PS Elements of General Net Theory // Lect. Notes in Computer Science. – 1980. – Vol. 84 – P. 21-163.
7. *Genrich H.J., Lautenbach K.* The Analysis of Distributed Systems by Means of Predicate / Transition Nets // Lect. Notes in Computer Science. – 1978. – Vol. 70 – P. 123-146.
8. *Hajek J.* Automatically Verified Data Transfer Protocol // Proc. 4 th Int. Comput. Comm. Conf. (Kyoto, Japan, Sept., 1978). – P. 749-756.
9. *Пранявічус Г.І.* Моделі і методи дослідження обчислювальних систем. – Вільнюс: Монслас, 1982. – 228 с.
10. *Тіммерсон Дис.* Теорія мереж Петрі і моделювання систем. – М.: Світ, 1984. – 263 с.
11. *Боцман Г.В., Мерлін Ф.М.* Розробка зв'язкових протоколів // Проблеми МСНТІ. – М.: МЦНТИ, 1981. –№ 2. - С. 146-155.
12. *Андерсон Р.* Доказ правильності програм. – М.: Світ, 1982. – 163 с.
13. *Schwartz R., Melliaz- Smith P.M.* From State Machines to Temporal Logic Specification Methods for Protocol Standards // IEEE Trans. Comm.– 1982. – V.30, № 15. – P. 2486-2496.
14. *Hack M.* Analysis of Production Schemata by Petri Nets//Technical Report 94:Project HAC, MIT. – Cambrige, 1972. – 119p.
15. *Якубайтис Э. А.* Информационно- вычислительные сети. – М.: Финансы и статистика, 1981. – 256с.
16. ISO/ TC 97/SC 21 N1528 ISO/DP 7498/ 2. Information Processing Systems – OSI Reference Model – Part 2: Security Architecture.
17. ISO/ TC 97/SC 21 N1553 ISO/DP 7493/ 3. Information Processing Systems – OSI Reference Model – Part 3: Naming and Addresing.
18. *Berthlot G., Terrat R.* Petri Nets Theory for the Correctness of Protocols // IEEE Trans. – 1982. – Vol. Com 30, № 12 – P. 2497-2505.
19. *Keller R.M.* Formal Verification of Parallel Programs // Comm. ACM. – 1976. – V. 19, № 17. – P. 371-384с.

Розділ 7.
Приклади задач

В додатках пропонуються питання, які можуть бути використані студентам для самостійного опрацювання для поглиблення знань, викладачами для організації семінарських занять, курсового проектування і іншого.

7.1. Теми завдань для самостійного виконання

Пропонуються наступні теми завдань:

– для заданих структур мов програмування побудувати системи граматичних продукцій у вигляді синтаксичних діаграм,
зауваження – представлення синтаксичних діаграм дивися на рис. 4.7;

– для заданих структур мов програмування побудувати конструктивні граматичні структури з продукціями у вигляді нотацій Бекуса – Наура,
зауваження – необхідно розглянути п. 5.7.1;

– для заданих структур мов програмування побудувати символьні системи рівнянь і знайти їх розв’язки,
зауваження – рекомендується розглянути п. 5.7.2 і [2];

– для запропонованих програм з’ясувати їх структурну еквівалентність;
зауваження – слід скористатися схемою алгоритму для встановлення структурної еквівалентності програм, яка наведена у п. 3.4 і роботі [3];

– створити структурну модель запропонованої БД;
зауваження – скористатися роботами [1] і [7];

– знайти метричні виміри характеристик програм,
зауваження – спочатку для програми побудувати структурний граф, прийнявши за його вершини оператори або строки про-

грами, а дуги – зв'язки між операторами і далі знайти виміри графу (див. пп. 3.2 – 3.5); при програмній реалізації вимірів зручно використовувати матрицю суміжностей графу;
– знайти часові трудовитрати запропонованих програм, *зауваження* – скористуватися методом Кірхгофа для часового виміру програм [5, 6];
– встановити ефективність використання пам'яті програмами, *зауваження* – скористуватися схемами програм Лаврова [6];
– встановити правильність запропонованої програми; *зауваження* – скористуватися схемами програм Лаврова [6];
– створити моделі регулярних програм за заданими алгоритмічними програмами.

зауваження – представити алгоритмічну програму автоматом і за його графом переходів записати регулярну програму на наборі операторів алгебри Глушкова (див. п. 4.3.1 і [2]).

7.2. Приклади вирішення завдань

7.2.1. Задача структурної еквівалентності програм

Предметна постановка задачі

Задача структурної еквівалентності розв'язується за схемою п. 3.4, за якою для заданих програм $\mathfrak{R}_1$ і $\mathfrak{R}_2$ спочатку будуються структурні графи. Так як процес побудови графів – елементарний, наприклад, за вершини графу прийняті оператори (з врахуванням їх повторів), за дуги функціональні зв'язки між операторами програми, тому алгоритмічні програми не приводимо. Припустимо, що програмам $\mathfrak{R}_1$ і $\mathfrak{R}_2$ відповідають графи G_1 і G_2 зображені на рис. 7.1.

Необхідно встановити ізомофність неорієнтованих графів G_1 і G_2 та еквівалентність програм $\mathfrak{R}_1$ і $\mathfrak{R}_2$.

▷ Аналіз кількості вершин графів (кількість однакова) свідчить про те, що графи є претендентами на ізоморфність. Тому далі підрахуємо степені вершин графу G_1: $st(c_1)=3$, $st(c_2)=3$, $st(c_3)=4$, $st(c_4)=2$, $st(c_5)=5$ і $st(c_6)=3$. За значеннями j степенів вершини цього графу утворюють чотири класи K_j, $j=2,3,4,5$ так, що $K_2=\{c_4\}$, $K_3=\{c_1,c_2,c_6\}$, $K_4=\{c_3\}$ і $K_5=\{c_5\}$. Аналогічно для вер-

шин другого графу маємо: $S_2 = \{v_6\}$, $S_3 = \{v_1, v_3, v_4\}$, $S_4 = \{v_2\}$ і $S_5 = \{v_5\}$. Так, як кількість класів, їх степеневі розміри і кількості елементів у відповідних класах однакові, тому графи G_1 і G_2 залишаються претендентами на ізоморфність. Але групи класів K_2, K_4, K_5 і S_2, S_4, S_5 мають однакові відповідні степені вершин та по одному елементові у класах, що свідчить про існування взаємно однозначного бінарного відношення ρ між вершинами класів з однаковими степеневими значеннями. Отже відношення встановлюється між вершинами $\rho(c_4, v_6)$, $\rho(c_3, v_2)$ і $\rho(c_5, v_5)$.

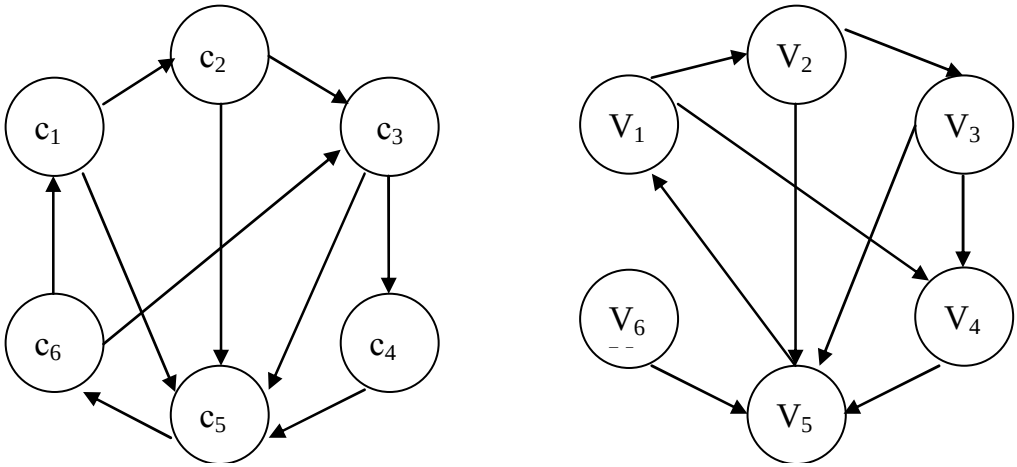


Рис. 7.1. Структурні графи G_1 і G_2

Зостається не з'ясованим питання відношення на елементах класів K_3 і S_3. Вирішити це питання можна за допомогою процесу розбиття класів K_3 і S_3 на підкласи (якщо можливо). Процес розбиття виконується доти поки у підкласах зостанеться по одній вершині. Розбиття спирається на розрахунки кількості зв'язків між вершинами класу K_3 (S_3) з елементами відповідних груп класів. Результати розрахунків зручно представляти матрицею, строки якої є класи відповідних груп, стовбці вершини класу розбиття. У клітинах вказується кількість зв'язків вершини стовпчика з елементами класів відповідних строк.

Виконаємо перший крок процесу розбиття класів K_3 і S_3.

$$
\begin{array}{c} \\ K_2 \\ K_3 \\ K_4 \\ K_5 \end{array}
\begin{array}{c} \begin{array}{ccc} c_1 & c_2 & c_6 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 2 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \end{array},
\quad
\begin{array}{c} \\ S_2 \\ S_3 \\ S_4 \\ S_5 \end{array}
\begin{array}{c} \begin{array}{ccc} v_1 & v_3 & v_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 2 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \end{array}..
$$

Аналіз змісту стовбців першої матриці показує, що перший стовбець відрізняється від другого і третього, а змісти другого і третього однакові, тому клас K_3 розбивається на підкласи $K_3' = \{c_1\}$ і $K_3'' = \{c_2, c_6\}$. Аналогічно маємо для класу S_3 – $S_3' = \{v_1, v_3\}$, $S_3'' = \{v_4\}$. Порівнюючи результати розбиття, з'ясовуємо, що стовбці матриці для класів K_3', S_3'' і K_3'', S_3' однакові, але для елементів класів K_3' і S_3'' існує відношення $\rho(c_1, v_4)$. Класи K_3'', S_3' потребують подальшого розбиття. Результат спроби розбиття наведено у матрицях

$$
\begin{array}{c} \\ K_2 \\ K_3' \\ K_3'' \\ K_4 \\ K_5 \end{array}
\begin{array}{c} \begin{array}{cc} c_2 & c_6 \end{array} \\ \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \end{array},
\quad
\begin{array}{c} \\ S_2 \\ S_3'' \\ S_3' \\ S_4 \\ S_5 \end{array}
\begin{array}{c} \begin{array}{cc} v_1 & v_3 \end{array} \\ \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \end{array}.
$$

Розбити класи K_3'' і S_3' на підкласи не вдалося, однак однакові стовбці у матрицях свідчить про існування альтернативних відношень як для вершини c_2, так і для – c_6. Тобто $\rho(c_2, v_1)$ і $\rho(c_6, v_3)$ або $\rho(c_2, v_3)$ і $\rho(c_6, v_1)$. Отже графи G_1 і G_2 (як неорієнтовані) ізоморфні.

З'ясуємо тепер орієнтацію дуг на ізоморфних вершинах. Так на кожній вершині ізоморфної пари (c_4, v_6) (див. рис. 7.1) маємо одну вхідну і одну вихідну дуги; на парі (c_3, v_2) – по дві вхідні і вихідні дуги; для пари (c_5, v_5) порушується однозначність орієнтації тому, що у вершину c_5 входить чотири дуги і одна – виходить, а на вершині v_5 маємо три входи і два виходи. Отже, програми, яким відповідають структурні графи G_1 і G_2 не еквівалентні. ◁

Якщо ж програмам $\Re_1$ і $\Re_2$ відповідають структурні графи G_3 і G_4 зображені на рис. 7.2, тоді можна переконатися, що всі ізоморфні пари вершин мають однозначну відповідність по орієнтації дуг і – програми будуть структурно еквівалентними.

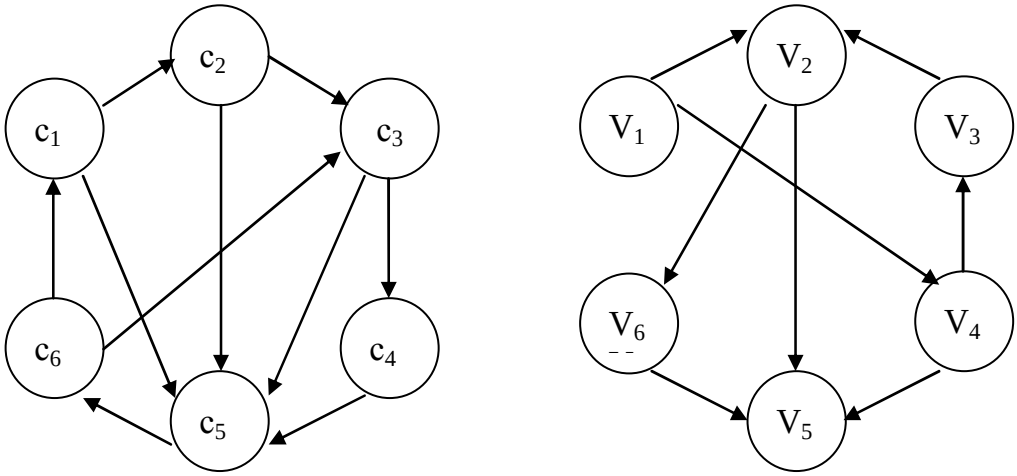


Рис. 7.2. Структурні графи G_3 і G_4

7.2.2. Задача ефективного використання пам’яті програмою

Предметна постановка задачі

Нехай задана програма $\Re$, зміст якої є

```
void Um_Razm()
{
  w=0; zn=0;
  mk[0][0]="[";
  k=1;
  for(i=0;i<n;i++)
    {for(j=0;j<n;j++)
        {mk[k][0]=matr[i][j]; k++;}
    mk[k][0]="|";
    k++;
    }
  mk[k-1][0]="]";
  razm--;
  Det(razm,0,k);
}
```

За методом Лаврова з’ясувати ефективність використання пам'яті цією програмою.

▹ Запропонована задача відноситься до задач аналізу властивостей алгоритмічних програм, їх перетворення з метою економії пам'яті. Аналіз використання пам'яті програмою здійснюється за допомогою операторних схем Лаврова [5, 6]. Операторна схема може задаватися графічно як набір операторних блоків (вершин графу) і дуг-переходів, відтворюючих логіку алгоритмічної програми. Операторні блоки навантажуються вхідними і вихідними вказівниками змінних, які використовуються і створюються операторами блоків. Сформуємо операторну схему для наведеної програми. Для цього спочатку створимо таблицю імен-позначок операторних блоків. Іменування блоків може бути за окремими операторами (табл.2) або за групами однакових операторів (табл. 7.1).

Таблиця 7.1

Імена блоків за групами операторів програми

Ім’я блоку	Зміст блоку
Ім’я блоку	Зміст блоку
c_1	w=0; zn=0; mk[0][0]=”[“; k=1; i=0;
c_2	for(умова){тіло};
c_3	j=0;
c_4	for(умова){тіло};
c_5	mk[k][0]=matr[i][j];
c_6	k++; j++; i++;
c_7	mk[k][0]=”|”;
c_8	k++;
c_9	mk[k-1][0]=”]”;
c_{10}	razm--;
c_{11}	

Таблиця 7.2

Імена блоків за окремими операторами програми

Ім’я блоку	Зміст блоку
s_1	w=0;
s_2	zn=0;
s_3	mk[0][0]=”[“
s_4	k=1
s_5	i=0
s_6	for(умова){тіло}
s_7	j=0
s_8	for(умова){тіло}
s_9	mk[k][0]=matr[i][j]
s_{10}	k++
s_{11}	j++
s_{12}	i++
s_{13}	mk[k][0]=”|”
s_{14}	k++
s_{15}	mk[k-1][0]=”]”
s_{16}	razm--;
s_{17}	Det(razm,0,k);

Для проведення аналізу з використання пам’яті програмою $\Re$ зручно використовувати операторну схему (рис. 7.3) побудовану на основі табл. 7.2.

Виконуючи дослідження операторної схеми знаходимо життєві циклів змінних програми. У табл. 7.3 наведені границі життєвих циклів змінних програми.

Аналіз даних табл. 7.3 показує, що життєві цикли змінних w, zn та matr обмежені одним блоком, але для змінних w і zn їх життєві цикли не покриваються циклами інших змінних. Тому змінні w і zn можуть бути видалені із програми. В результаті подальшого аналізу даних таблиці з’ясовується, що життєві цикли змінної razm не покриваються відповідними циклами змінних i , n та j , виходячи з цього замість імені razm можна використати одне із імен i , n або j. Отже для програми $\Re$ мається можливість скотити пам'ять на три одиниці. $\triangleleft$

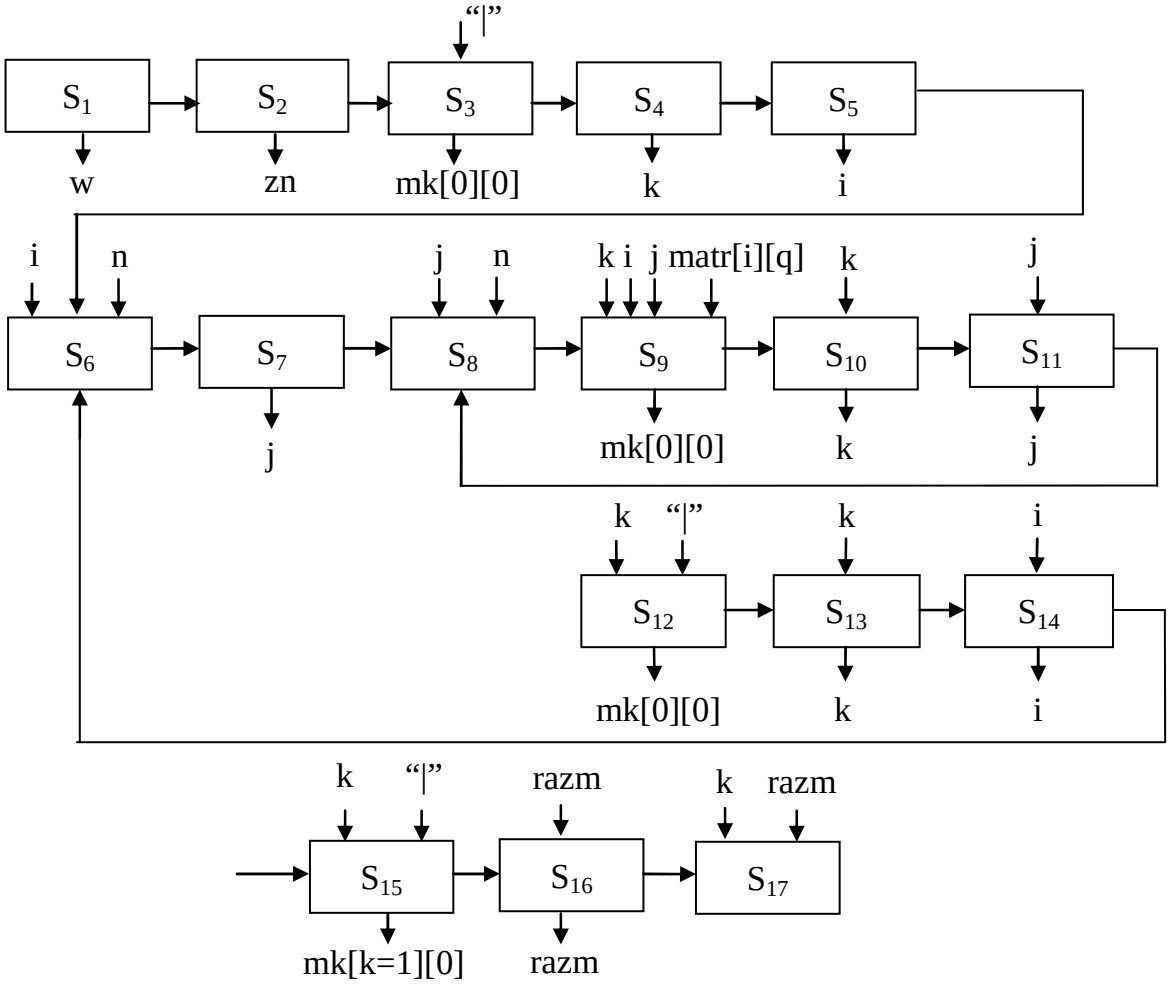


Рис. 7.3. Операторна схема програми $\Re$ з навантаженими вершинами

Таблиця 7.3

Границі життєвих циклів змінних програми $\Re$

змінна	початкова вершина	кінцева вершина
w	s_1	s_1
zn	s_2	s_2
mk	s_3	s_{15}
k	s_4	s_{17}
i	s_5	s_{12}
n	s_6	s_8
j	s_7	s_{11}
matr	s_9	s_9
razm	s_{16}	s_{17}

7.2.3. Розрахунок часових трудовитрат алгоритмічних програм

Предметна постановка задачі

Для програми $\Re$ підрахувати часові трудовитрати реалізації алгоритму, при умові, що оператор O_i програми виконується за час t_i.

▷ Часові трудовитрати алгоритму за програмою його реалізації можна визначити за різними методами, наприклад, за допомогою випадкових процесів, балансових відношень Кірхгофа і ін. Виходячи з того, що у попередніх розділах елементи ймовірностей не розглядалися, розглянемо підхід запропонований Кнутом [4]. Такий підхід передбачає виконання балансового відношення по кількості вхідної та вихідної інформації для кожного оператора O_i програми і підрахування при цьому кількості реалізацій n_i оператора. Тоді часові витрати алгоритму розраховуються за формулою

$$T = \sum_i n_i t_i . \tag{7.1}$$

Для знаходження кількості реалізацій операторів програми за пропозицією Лаврова [6] слід застосувати операторну схему програми, навантаживши її дуги невідомими кількостями проходів дуг g_j. Замкненість графу операторної схеми досягається введенням віртуальної дуги, яка з'єднує заключну і початкові вершини графу з позначкою g_0, із значенням 1. Тепер, застосовуючи балансове відношення на i-вершині графу, отримаємо

$$_i\sum_{вх} g_j = {}_i\sum_{вих} g_j \tag{7.2}$$

і знайдемо кількість реалізацій $n_i = {}_i\sum_{вх} g_j$ або $n_i = {}_i\sum_{вих} g_j$.

Застосуємо цю методику до програми $\Re$. Навантажена операторна схема за позначеннями блоків (табл. 7.2) зображена на рис. 7.4.

За графом (рис. 7.4) і виразом (7.2) побудуємо систему рівнянь відносно невідомих g_j, додавши до неї умову $g_0 = 1$.

$$\begin{array}{lll}
g_0 = 1, & g_5 + g_{15} = g_6 + g_{16}, & g_{14} = g_{15}, \\
g_0 = g_1, & g_6 = g_7, & g_{12} = g_{13}, \\
g_1 = g_2, & g_7 + g_{11} = g_8 + g_{12}, & g_{13} = g_{14}, \\
g_2 = g_3, & g_8 = g_9, & g_{16} = g_{17} \\
g_3 = g_4, & g_9 = g_{10}, & g_{17} = g_{18}, \\
g_4 = g_5, & g_{10} = g_{11} & g_{18} = g_0.
\end{array}$$

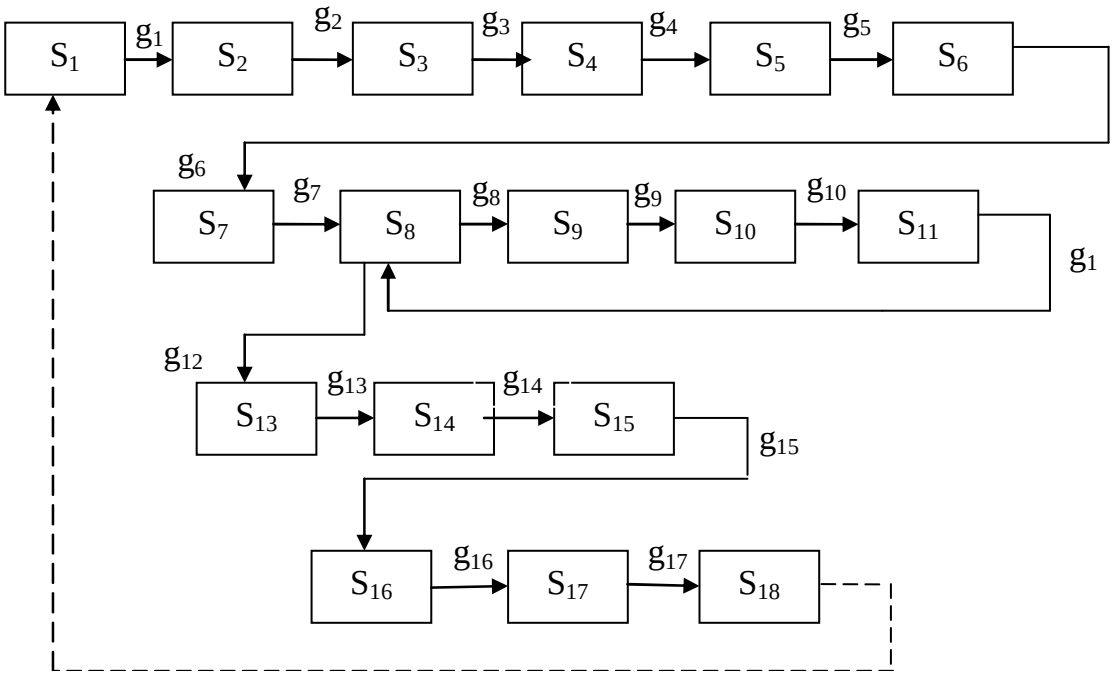


Рис. 7.4. Операторна схема програми $\Re$ з навантаженими дугами

Система рівнянь є недовизначеною, тому що кількість невідомих – 19, а кількість рівнянь – 18. Отже система має безліч розв'язків. Для розв'язання лінійної системи можна використати прийом Кіргофа [6] або застосувати методи алгебри. Застосовуючи алгебраїчний прийом виключення невідомих, знаходимо, що: $g_1 = g_2 = g_3 = g_4 = g_5 = 1$ і $g_{18} = g_{17} = g_{16} = 1$, а змінні $g_{15} = g_{14} = g_{13} = g_{12} = g_7 = g_6$ і $g_9 = g_{10} = g_{11} = g_8$. Придаючи довільним невідомим g_6 і g_8 базисні значення: 0 або 1, отримаємо при $g_6 = 1$ та $g_8 = 0$ значення кількості реалізацій $n_6 = 2$ та $n_8 = 1$, а при $g_6 = 0$ та $g_8 = 1$ значення кількості реалізацій будуть $n_6 = 1$ та $n_8 = 1$. Таким чином для варіантів розгалуження у вершинах g_6 і g_8 часові витрати будуть: $T_{0,1} = \sum_{i=1}^{17} t_i + 2t_6$, $T_{1,0} = \sum_{i=1}^{17} t_i$, де у першому виразі $i = 6$ під знак суми не входить.

Література до розділу 7

1. *Гаврилова Т А., Хорошевский В Ф.* Базы знаний интеллектуальных систем. – СПб: Питер, – 2000. – 384 с.
2. *Глушков В. М., Цейтлин Г. Е., Ющенко Е. Л.* Алгебра. Языки. Программирование. – К.: Наукова думка, 1989. – 376 с.
3. *Горбатов В. А.* Основы дискретной математики. М.: Высш. шк., 1986. – 311 с.
4. *Кнут Д.* Искусство программирования, том 1. Основные алгоритмы, 3-е изд. – М.: Издательский дом «Вильямс», 2000. – 720 с.
5. *Котов В. Е., Сабельфельд В. К.* Теория схем программ. М.: Наука, 1991. – 248 с.
6. *Лавров С.С.* Программирование. Математические основы, средства, теория. – СПб.: БХВ-Петербург, – 2001. – 320 с.
7. Представление и использование знаний. / Под ред. Х. Уэно, М. Исидзуки. – М.: Мир, 1989. – 220 с.

ЗМІСТ

Навчальне видання

Скалозуб *Владислав Васильович*
Ільман *Валерій Михайлович*
Івченко *Юрій Миколайович*
Андрющенко *Вадим Олександрович*

Дискретні та алгоритмічні структури в інструментарії програмної інженерії

Навчальний посібник

Редактор *О. О. Котова*
Комп'ютерна верстка і дизайн *В. І. Шинкаренко*

Формат 60×84 $\frac{1}{16}$. Ум. друк. арк. 14,82.
Обл.-вид. арк. 14,90. Наклад 300 пр. Зам. №

Дніпропетровський національний університет
залізничного транспорту імені академіка В. Лазаряна
Свідоцтво суб'єкта видавничої справи ДК № 1315 від 31.03.2003.

Адреса видавця та дільниці оперативної поліграфії:
вул. Лазаряна, 2, м. Дніпропетровськ, 49010.