3.4 Алгоритмічні моделі

Бурхливий розвиток комп'ютерної техніки, проникнення її в усі сфери сучасної цивілізації привернули особливу увагу до тих розділів математичного моделювання, які є теоретичною основою методів комп'ютеризації і програмування. Такою основою є, в першу чергу, теорія алгоритмів.

Теорію алгоритмів можна поділити на класичну і прикладну. У класичній теорії алгоритмів існує безліч формальних методів опису алгоритмів. Серед них можна виділити найзначніші: арифметичне числення предикатів Гьоделя, машини Поста і Тюрінга, автомати Маркова, схеми Янова, блок-схеми. У прикладній теорії алгоритмів розглядаються способи розв'язання прикладних алгоритмічних задач, методи оцінювання характеристик прикладних алгоритмів (складність, збіжність тощо).

3.4.1 Основні поняття теорії алгоритмів

Поняття ***алгоритму*** є одним з базових понять математики (таких, як число, множина, точка тощо), про які існує лише інтуїтивне уявлення без строгого математичного означення, оскільки будь-яке означення само ґрунтується на базових поняттях.

Інтуїтивне поняття алгоритму містить у собі кілька загальних рис:

1. *Алгоритм* — це процес послідовного отримання результату, який проходить у дискретному часі (*дискретність алгоритму*);

2. Результат, що одержується у певний, відмінний від початкового момент часу, однозначно визначається системою величин, одержаних у попередні моменти часу (*детермінованість алгоритму*);

4. Якщо спосіб одержання наступної величини з якої-небудь заданої величини не дає результату, то слід вказати, що вважається результатом алгоритму (*результативність алгоритму*);

5. Початкова система величин може вибиратись із деякої потенційно нескінченної множини (*масовість алгоритму*).

Алгоритми підрозділяються на декілька типів (рис. 3.17).

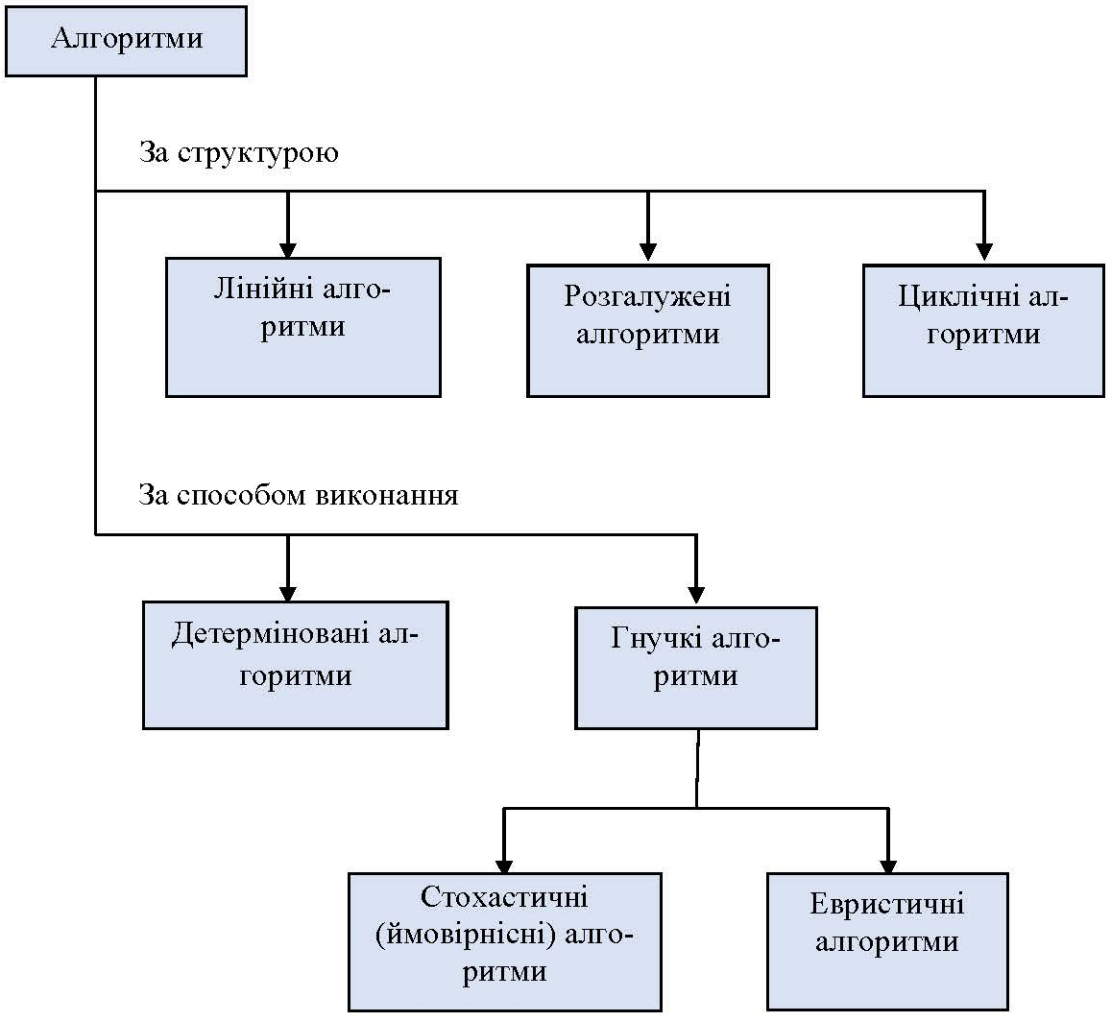


Рисунок 3.17 – Загальна класифікація алгоритмів

Детерміновані, жорсткі алгоритми (наприклад, алгоритм роботи машини, двигуна і т. п.) задають визначені дії, позначаючи їх у єдиній і достовірній послідовності, забезпечуючи тим самим однозначний результат, якщо виконуються ті умови процесу, задачі, для яких розроблений алгоритм.

Ймовірнісний (стохастичний) алгоритм використовує випадкові дані, що виробляються спеціальним генератором, а результати отримуються після статистичної обробки. До стохастичних алгоритмів відносять, наприклад, різноманітні методи випадкового пошуку.

Евристичний алгоритм (еври́стика (грец. ευρίσκω (heuristiko) — знаходжу, відшукую, відкриваю) – це такий алгоритм, у якому досягнення кінцевого результату програми дій однозначно не визначено, так само як не позначена вся послідовність дій, не виявлені всі дії виконавця. До евристичних алгоритмів від-

носять, наприклад, інструкції і розпорядження. У цих алгоритмах використовуються універсальні логічні процедури і способи прийняття рішень, засновані на аналогіях, асоціаціях і минулому досвіді розв'язання схожих задач.

За структурою алгоритми поділяються на лінійні, розгалужені і циклічні.

Лінійний алгоритм – набір команд (вказівок), які виконуються послідовно в часі одна за одною.

Розгалужений алгоритм – алгоритм, що містить хоча б одну умову, в результаті перевірки якої ЕОМ забезпечує перехід на один із двох можливих кроків.

Циклічний алгоритм – алгоритм, що передбачає багаторазове повторення тієї самої дії (тих самих операцій) над новими вихідними даними. До циклічних алгоритмів зводиться більшість методів обчислень, перебору варіантів.

Алгоритмічна модель – це запис алгоритму функціонування системи у певний спосіб. Таким чином, поняття алгоритмічної моделі є узагальненням поняття алгоритму і його застосуванням для моделювання ширшого класу дискретних процесів і систем, ніж обчислювальні.

Кожен з способів запису алгоритмів має переважну сферу застосування. Найбільш поширені такі форми запису алгоритмів:

– *формальна* (запис за допомогою літерно-цифрових позначень та додаткових символів у вигляді формул). Формальний спосіб застосовується у теоретичних дослідженнях алгоритмів як математичних об'єктів;

– *словесна* (запис природною мовою). Словесний спосіб використовується у різноманітних інструкціях для персоналу;

– *графічна* (зображення за допомогою графічних символів). Графічний спосіб характеризується високою наочністю, тому використовується при необхідності пояснення та евристичного аналізу алгоритму;

– *псевдокоди* (напівформалізовані описи алгоритмів умовною алгоритмічною мовою, що містять у собі як елементи мови програмування, так і фрази природної мови, загальноприйняті математичні позначення тощо). Псевдокоди використовуються при записі алгоритмів, які призначені для реалізації на комп'ютері, але заздалегідь невідомо, яка мова програмування буде обрана для цієї реалізації;

– *програмна* (тексти мовами програмування). Програмний спосіб використовується при реалізації алгоритму на комп'ютері.

У загальному розумінні алгоритм є описом послідовності дій, в результаті якої з визначеного набору початкових даних (вхідних впливів, ресурсів тощо) X буде отриманий певний результат Y

$$X \xrightarrow{A} Y. \tag{3.90}$$

Вираз (3.90) є відображенням одної множини на іншу, отже, алгоритм можна розглядати як узагальнений оператор перетворення множини X на множину Y. Певна відмінність, яка вирізняє алгоритми від інших типів операторів, полягає у їх потенційно необмеженій складності – наразі довжина запису алгоритму

мовою програмування може складати десятки мегабайтів, а отримання результату може вимагати до 10^{10}–10^{11} елементарних операцій (порівняйте: час існування нашого Всесвіту складає приблизно $4 \cdot 10^{16}$ секунд)!

Алгоритми, автомати і рекурсивні функції

Теорія алгоритмів вивчає основні закономірності роботи алгоритмів, способи доведення їх правильності, способи визначення, чи є у алгоритмічної задачі розв'язок. З іншого боку, теорія алгоритмів займається пошуком алгоритмічно нерозв'язних проблем.

Існує декілька напрямків у теорії алгоритмів, які відрізняються так само, як відрізняються між собою різні алгебри, тобто переліком об'єктів, з якими працюють алгоритми, та операцій, які можуть виконуватися над цими об'єктами.

Основними напрямками теорії алгоритмів є *алгебра рекурсивних функцій* і *теорія автоматів*.

В теорії алгоритмів доведений ізоморфізм цих теорій, тобто, якщо результат отримано за допомогою, наприклад, рекурсивних функцій, то аналогічний результат обов'язково може бути отримано за допомогою теорії автоматів і навпаки. Цей ізоморфізм сформульований англійським вченим Черчем.

Останнім часом цікаві результати в теорії алгоритмів отримали за допомогою алгоритмічної алгебри Глушкова.

Скінченний автомат є перетворювачем, вихід якого залежить не тільки від вхідних і вихідних сигналів, але й від поточного стану автомата, причому кількість вхідних і вихідних змінних, кількість можливих значень цих змінних, а також кількість можливих станів автомата скінченна. Поточний стан автомата зберігається в його пам'яті.

Елементарні автомати можна розглядати як машини, що мають пам'ять у вигляді стрічки та пристрій для зчитування інформації зі стрічки. Стрічка розділена на сегменти, в яких розміщуються певні символи – команди і дані. Автомат, проглядаючи ці сегменти по черзі, виконує окремі обчислення і розв'язує задачі.

Елементарні автомати мають досить обмежені обчислювальні можливості. Більше можливостей мають автомати з нескінченною пам'яттю магазинного типу, але й вони є недостатньо універсальними. Найбільш універсальною моделлю комп'ютерних обчислень є машина Тьюрінга.

Машина Тьюрінга – математичне поняття, введене для формального уточнення інтуїтивного поняття алгоритму. Названа на честь англійського математика Алана Тюрінга, який і запропонував це поняття в 1936 р. Аналогічну конструкцію машини згодом і незалежно від Тьюрінга ввів американський математик Еміль Пост.

У кожної машини Тьюрінга є стрічка, потенційно нескінченна в обидві сторони (рис. 3.18).

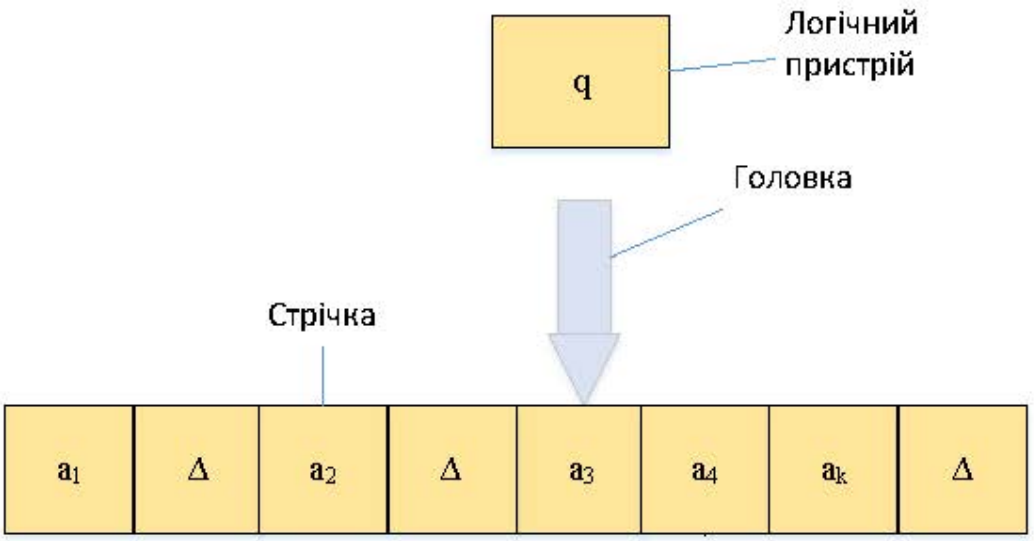


Рисунок 3.18 – Машина Тьюрінга

Є скінченна множина символів стрічки S_0, …, S_n, що називається ***алфавітом машини***. У кожний момент часу кожна комірка може бути зайнята не більш ніж одним символом. Машина має деяку скінченну множину внутрішніх станів q_0, q_1, …, q_n. У кожний момент часу машина знаходиться в тільки одному із цих станів.

Нарешті, є головка, яка у кожний момент часу знаходиться на одній із комірок стрічки. Машина діє не неперервно, а лише в дискретні моменти часу. Якщо в якийсь момент t головка сприймає комірку (тобто знаходиться на комірці), що містить символ S_i, і машина знаходиться у внутрішньому стані q_j, то дія машини визначена.

Тьюрінг показав, що будь-який *алгоритм може бути реалізований відповідною машиною Тьюрінга.*

Машина Поста (МП) – абстрактна обчислювальна машина, запропонована Емілем Леоном Постом, що відрізняється від машини Тьюрінга більшою простотою.

МП складається з головки і розбитої на секції нескінченної в обидва боки стрічки. Кожна секція стрічки може бути або порожня – 0, або позначеною міткою 1. За один крок головка може зрушитися на одну позицію вліво або вправо, поставити або знищити символ у тому місці, де вона стоїть. Робота машини Поста визначається програмою, що складається з кінцевого числа рядків.

Машини Тьюрінга і Поста еквівалентні з точки зору відображення поняття "алгоритм".

3.4.2 Основи алгоритмічної алгебри

Алгебру алгоритмів запропонував В. М. Глушков у 1965 р.

Алгебра алгоритмів – система, яка складається з двох алгебр u та v, які називаються відповідно алгеброю операторів та алгеброю умов. Елементи алгебри u – це перетворення (оператори) деякої абстрактної множини B, а елементи алгебри v – умови (предикати), визначені на множині B. Алгебру алгоритмів використовують для опису перетворень, виконуваних дискретними перетворювачами. У цьому випадку множина B називається інформаційною множиною.

Основна операція алгебри u – це звичайна операція множення (суперпозиції) операторів. Крім цієї операції, для кожної умови β з v в алгебрі u визначаються ще дві операції, називані β-диз'юнкцією і β-ітерацією операторів. Результатом β-диз'юнкції двох операторів P и Q є оператор R такий, що для будь-якого стану $b \in B$, $bR = bP$, якщо умова β істинна на стані b, $bR = bQ$, якщо $\beta(b)$ хибна й, нарешті, оператор R вважається невизначеним на стані b, якщо $\beta(b)$ не визначено. Результатом β-ітерації $\{P\}_\beta$ оператора P є оператор Q такий, що для будь-якого $b \in B$ має місце $bQ = bP^n$, де n — найменше із чисел m=0, 1, ... таких, що $\beta(bP^m)$ істинно ($bP^0 = b$ для будь-якого оператора P).

На множині v умов визначені звичайні булеві операції $\wedge$, $\vee$, $\neg$. Наприклад, диз'юнкція $\alpha \vee \beta$ двох умов є новою умовою γ. Ця умова приймає значення «1» на тих елементах множини B, на яких одна з умов α або β приймає значення «1». Значення «0» вона приймає на тих елементах, на яких α та β дорівгюють «0», і не визначена, якщо одна з умов α та β не визначена, а інша дорівнює «0». Крім цих операцій визначається операція $P \cdot \alpha$ множення оператора на умову. Результатом виконання цієї операції є умова β, значення якої дорівнює значенню умови α після виконання оператора P.

Базові алгоритми

Переважна більшість задач і алгоритмів може бути зведена до використання комбінації невеликої кількості типових базових алгоритмів. До таких слід віднести:

- послідовне виконання;
- вибір варіанта дій за умовою;
- циклічне виконання;
- рекурсія;
- ітерація;
- вибір одного значення з множини.

Для реалізації цих прийомів у мовах програмування передбачені спеціальні оператори та засоби.

Лінійні алгоритми (послідовне виконання) використовуються, наприклад, для обчислення значення функції з необмеженою областю визначення та для інших задач із заздалегідь визначеною послідовністю розв'язання.

Алгоритми з розгалуженням (вибір варіанта дій за умовою) використовуються, наприклад, для обчислення значення функції з обмеженою областю

визначення та для інших задач, при вирішенні яких виникає необхідність перевірки умов, а також у випадках вибору варіанта з декількох можливих.

Циклічні алгоритми використовуються при необхідності багаторазового виконання одних і тих же дій.

Рекурсія – це фундаментальне поняття, яке означає покрокове отримання результату, причому на наступному кроці використовується результат, отриманий на попередньому кроці.

Рекурсія може бути отримана трьома способами.

1) Циклічний виклик оператора $x:=f(x)$, де змінній x у лівій частині присвоюється нове значення, отримане за допомогою деякої функції f від старого значення.

2) Визначення рекурсивної функції, якщо така можливість передбачена мовою програмування.

У вузькому програмістському розумінні рекурсія – це виклик у процедурі або функції її самої. Головний момент у побудові рекурсивної програми – визначення умови припинення рекурсії.

Ітерація – це процес поступового наближення до правильного розв'язку. Використовується найчастіше для розв'язання рівнянь і систем рівнянь.

Вибір необхідного даного, яке задовольняє певну умову, з набору даних залежить від способу організації цього набору, наприклад *вибір найменшого даного* з масиву, *вибір з файлу* рядків за заданою ознакою.

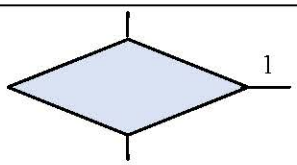
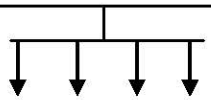
3.4.3 Подання алгоритмічних моделей

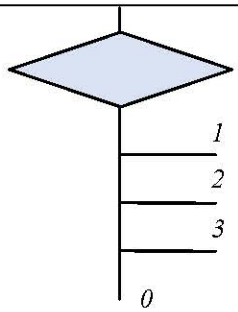
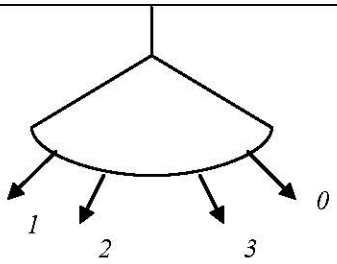
Зміст дій, що їх передбачає алгоритм, суттєво залежить від того, які процеси у системах описує алгоритмічна модель. Розрізняють алгоритми обробки даних (обчислювальні алгоритми), алгоритми вимірювання та збору вимірювальної інформації (алгоритми інформаційно-вимірювальних систем – ІВС), алгоритми передавання даних (протоколи зв'язку), алгоритми управління (в тому числі алгоритми автоматизованих систем управління – АСУ), алгоритми взаємодії з людиною-оператором (інтерфейси з оператором), алгоритми прийняття і узгодження рішень, алгоритми взаємодії груп операторів тощо.

Блок-схеми

Цей спосіб виявився дуже зручним засобом зображення алгоритмічних моделей і одержав широке поширення в науковій і навчальній літературі. Структурна схема алгоритму – графічне зображення алгоритму у вигляді схеми зв'язаних між собою за допомогою стрілок (ліній переходу) блоків – графічних символів, кожний з який відповідає одному кроку алгоритму. Основні символи блок-схем алгоритмів наведені у таблиці 3.2. Усередині блока дається опис відповідної дії.

Таблиця 3.2 – Основні символи блок-схем алгоритмів

Позначення	Зміст
	Процес
	Введення і виведення даних
	Прийняття рішення (перевірка умови)
	Виконання окремого алгоритму
	Виведення результатів на документ
	Початок і кінець алгоритму
	Цикл з параметром
	Лінія переходу
	Початок паралельних процесів
	Цикл з умовою (початок і кінець)
	Розрив алгоритму між сторінками і у межах сторінки

Позначення	Зміст
	Вибір автоматичний
	Вибір вручну

Принцип програмування “зверху донизу” вимагає, щоб блок-схема поетапно конкретизувалася і кожен блок «розписувався» до елементарних операцій. Але такий підхід можна здійснити при розв’язанні нескладних задач. При розв’язанні серйозної задачі блок-схема «розповзеться» до такого ступеня, що її неможливо буде охопити одним поглядом. Блок-схеми алгоритмів зручно використовувати для пояснення роботи вже готового алгоритму, при цьому як блоки беруться дійсно блоки алгоритму, робота яких не вимагає пояснень. Блок-схема алгоритму повинна служити для спрощення зображення алгоритму, а не для ускладнення.

Подання алгоритмічних моделей мережами Петрі

Мережі Петрі – математичний апарат для моделювання динамічних дискретних систем. Вперше описані Карлом Петрі у 1962 році. Мережа Петрі є дводольним орієнтованим графом, який складається з вершин двох типів – позицій і переходів, зв’язаних між собою дугами. Вершини одного типу не можуть бути з’єднані безпосередньо.

Позиція – це стан, в якому знаходиться система або певна її частина. В позиціях можуть розміщуватися мітки (маркери), здатні переміщуватися мережею.

Модель мережі Петрі використовують для зображення і аналізу причинно-наслідкових зв’язків у системі. Для прив’язки до певних моментів часу тих або інших переходів використовуються події. Подією називають спрацьовування переходу, при якому мітки з вхідних позицій цього переходу пересуваються у вихідні позиції. Стан системи формується в результаті реалізації локальних операцій, які називаються *умовами* реалізації подій. Умова має ємність:

– умова не виконана – ємність дорівнює нулю;
– умова виконана – ємність дорівнює одиниці;
– умова виконана з *n*-кратним запасом – ємність дорівнює n.

У зображеннях мереж Петрі (рис. 3.19) події і умови зображуються символами, які називаються *переходами* (вертикальними або горизонтальними смужками – "бар'єрами") і *позиціями* (кружками). Умови-позиції і переходи-події пов'язані відносинами залежності, які зображуються за допомогою орієнтованих дуг. Позиції, з яких виходять дуги даного переходу, називають вхідними позиціями. Позиції, до яких ведуть дуги від даного переходу, називають вихідними позиціями. Виконання умов зображується розміщенням відповідної кількості міток у позицію. Перехід дозволений, якщо є як мінімум одна вхідна мітка у кожній з його вхідних позицій.

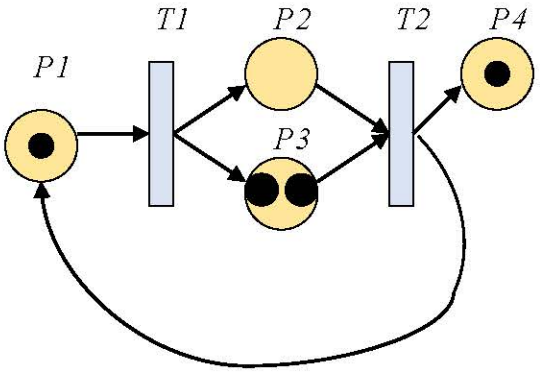


Рисунок 3.19 – Приклад мережі Петрі. Світлими кружками позначені позиції, смужками – переходи, чорними кружками – мітки

Мережі Петрі бувають таких видів:

– часова мережа Петрі – переходи мають вагу, яка означає час спрацьовування (затримки);
– стохастична мережа Петрі – затримки є випадковими величинами;
– функціональна мережа Петрі – затримки визначаються як функції деяких аргументів, наприклад, кількості міток у деяких позиціях, станів деяких переходів;
– кольорова мережа Петрі – мітки можуть бути різних типів, позначених кольорами, тип мітки може бути використаний як аргумент у функціональних мережах;
– інгібіторна мережа Петрі – можливі інгібіторні дуги, які забороняють спрацьовування переходу, тощо.

Основними властивостями мереж Петрі є:

– обмеженість – кількість міток у будь-якій позиції не може перевищити деякого значення K;
– безпека – окремий випадок обмеженості, K=1;

– збереженість – постійність ресурсів $\sum_i A_i N_i$, де N_i кількість маркерів в i-й позиції, A_i – ваговий коефіцієнт;

– досяжність – можливість переходу мережі з одного заданого стану, який характеризується певним розподілом міток, до іншого;

– живучість – можливість спрацьовування будь-якого переходу при функціонуванні об'єкта.

В основі аналізу моделі Петрі лежить аналіз досяжності станів, часу переходу з певного початкового стану у заданий тощо.

Змістовні логічні схеми алгоритмів

Для формалізованого опису алгоритмів роботи систем також використовується ***апарат змістовних логічних схем алгоритмів*** (ЛСА).

Крім множин функціональних $\{A_i\}$ і логічних $\{w_i\}$ операторів, які використовуються в алгоритмічній алгебрі, в ЛСА об'єднуються оператори, які визначають обмін інформаційними і службовими сигналами між функціональними блоками системи, а також перетворення цих сигналів. Окрім літерних позначень у ЛСА використовують символи, якими позначається порядок виконання операторів або структура апаратної частини системи. Основні елементи ЛСА у порівнянні з іншими способами запису алгоритмічних моделей наведені в таблиці 3.3.

До основних недоліків ЛСА можна віднести необхідність складання і постійного використання списків операторів $\{A_i\}$ і $\{w_i\}$ з розшифруванням їх змісту, відсутність ефективних методів мінімізації записів. Ці недоліки слабо проявляються при відносно простих і загострюються при складних ЛСА.

Перетворення алгоритмічної моделі формалізовані у вигляді алгебраїчної системи:

$$AS = (AM, OP), \tag{3.91}$$

де AM – множина алгоритмічних моделей; OP – множина операцій над ними.

Множина операцій OP складається з двох елементів:

– ***paste(B,n1,n2)*** – вставляння блока ***B*** в алгоритмічну модель між елементами з номерами ***n1*** і ***n2***;

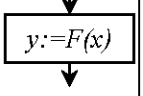
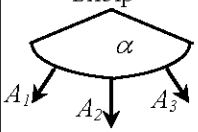
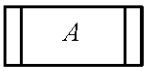
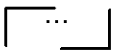
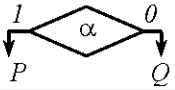
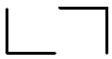
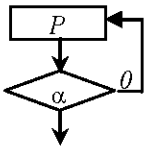
– ***cut(n1,n2)*** – вирізання блока з алгоритмічної моделі між елементами з номерами ***n1*** і ***n2***,

– а також понять одиничної операції **1**, яка не змінює моделі, і оберненої операції $\boldsymbol{op}^{-1}$, яка задовольняє умову

$$\boldsymbol{op}^{-1}(\boldsymbol{op}\,(AM)) = AM. \tag{3.92}$$

Еквівалентними перетвореннями алгоритмічної моделі будем називати таку послідовність операцій над моделлю, яка не змінює змісту результатів роботи системи (хоча можуть змінювати якісні показники як результатів, так і самої системи).

Таблиця 3.3 – Порівняння елементів і позначень алгоритмічних моделей

Елементи ЛСА		**Операції системи алгоритмічних алгебр**		**Типові елементи схем програм**	
Зміст	**Позначення**	**Зміст**	**Позначення**	**Зміст**	**Позначення**
Функціональний оператор	A_i	Оператор	$\tilde{u}$	функція	$F(x)$
Логічний оператор	w_i	Умова	$\hat{g}$	логічний вираз (функція)	L
Перетворення сигналів	x/y	Присвоювання		процес визначення значення	
Виконання будь-якого перетворення (ЧИ)	$A_1 \mid A_2 \mid A_3$			вибір 	case x of 1: A_1 2: A_2 3: A_3
Об'єднання перетворення	[.], { . }	Множення (композиція)	AB		Begin End
Виконання алгоритму програмним шляхом				окремо визначений алгоритм 	Procedure A
Перенос виконання алгоритму вперед		α-диз'юнкція	$\hat{g}_\alpha(P,Q)$	умовний перехід 	if α then P else Q
Перенос виконання алгоритму назад		α-ітерація $(\alpha\ \hat{g})$	$p\hat{g}_\alpha(P)$		repeat P until α

Еквівалентні перетворення здійснюються на основі властивостей алгебри *AS*:

1. $paste(B,n1,n2)\ cut(n1,n2) \equiv 1$
2. $cut(n1,n2)\ paste(B,n1,n2) \equiv 1$
3. $paste(B1,n1,n2)\ paste(B2,n3,n4) \equiv paste(B2,n3,n4)\ paste(B1,n1,n2)$ якщо $(n1,n2) \cap (n3,n4) = \varnothing$
4. $cut(n1,n2)\ cut(n3,n4) \equiv cut(n3,n4)\ cut(n1,n2)$ якщо $(n1,n2) \cap (n3,n4) = \varnothing$
5. $En1(op,X,Y)\ En2(op^{-1},Y,X) \equiv 1$

3.4.4 Ізоморфізм та гомеоморфізм в мовах програмування як основа комп'ютерного моделювання

Окремим і наразі найважливішим і найпоширенішим способом запису алгоритмічних моделей є комп'ютерні програми, які є поданням моделей алгоритмічною мовою.

Сучасні алгоритмічні мови є переважно універсальними (C++, Pascal, Ada, Java тощо), побудованими відповідно до стандарту ANSI. В свою чергу, стандарт ґрунтується на ідеях Ніколаса Вірта, автора мови Pascal. В основі цих ідей лежить реалізація у мові програмування основних алгебраїчних систем. Це знайшло своє відображення у понятті "типу даних", яке визначає певну множину даних (цілі числа, реальні або дробові числа, логічні дані, символи тощо) і набір операцій, які можна виконувати над даними кожної множини (наприклад, над цілими числами: додавання, віднімання, множення; над реальними числами: додавання, віднімання, множення, ділення; над логічними даними: кон'юнкція, диз'юнкція, інверсія). Такий підхід забезпечив відповідність комп'ютерних програм фундаментальним математичним законам.

З розвитком комп'ютерних технологій кінцевим етапом будь-якого способу математичного моделювання є програмна реалізація моделі та її дослідження на комп'ютері. Потенційна можливість такої реалізації є основою застосування комп'ютерів. Така можливість доведена Тьюрінгом, який показав, що за допомогою обмеженої кількості простих операцій, які належать певній скінченній множині, можна отримати розв'язок будь-якої алгоритмічно розв'язної задачі. На підставі цього можна стверджувати, для кожної задачі і будь-якої коректної математичної моделі знаходження розв'язку можна отримати ізоморфну програмну форму.

З розвитком мов програмування програмне забезпечення все більше відповідає суті математичного моделювання завдяки запровадженню принципів *об'єктно-орієнтованого програмування*. Введене поняття класу, під яким розуміють множину об'єктів, які характеризуються набором параметрів ("властивостей") і операцій ("правил", "методів") над ними. Поєднання в одному об'єкті параметрів і операцій, яке є запорукою математичної коректності моделювання, називають *інкапсуляцією*. Кожен програмний об'єкт фактично є моделлю деякого реального об'єкта, а взаємодія програмних об'єктів через механізми повідомлень, подій тощо є моделлю реальних процесів.

Технологія об'єктно-орієнтованого програмування дає спеціальний інструмент для створення гомеоморфних моделей систем. Цей інструмент називають властивостями спадкування та поліморфізму класів.

Спадкування – це можливість утворення нового класу, який містить в собі параметри і операції "батьківського" класу, а також додаткові параметри та операції. Між об'єктами "батьківського" і "дочірнього" класів існують відносини гомеоморфізму, оскільки одному батьківському класу може відповідати множина дочірніх класів.

Поліморфізм означає можливість заміни (переозначення) певної операції у дочірньому класі, що теж веде до відносин гомеоморфізму.