Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження процесу розробки моделі програмної системи методами теорії автоматів та мереж Петрі
(тема)

Виконав:
студент (ка) 2 курсу, групи ІПЗм-22-6

Пархоменко Б. Є.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник доц. кафедри ПІ Лановий О. Ф.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Пархоменку Богдану Євгеновичу
(прізвище, ім’я, по батькові)

1. Тема роботи Дослідження процесу розробки моделі програмної системи методами теорії автоматів та мереж Петрі

Затверджена наказом по університету від 29.03.2024р. № 250 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18.06.2024

3. Вихідні дані до роботи теорія множин, теорія мереж Петрі, теорія алгоритмів

4. Перелік питань, що потрібно опрацювати в роботі

теоретичні основи теорії автоматів та мереж Петрі, обґрунтування вибору інструментів моделювання, використання мереж Петрі для створення моделей алгоритмів, опис програмної системи

КАЛЕНДАРНИЙ ПЛАН

Номер	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Видача завдання	29.04.2024	виконано
2	Аналіз предметної галузі	01.05.2024	виконано
3	Постановка задачі	02.05.2024	виконано
4	Експериментальні дослідження	02.05 – 20.05.24	*виконано*
5	Аналіз результатів експериментальних досліджень та розробка рекомендацій	20.05 – 22.05.24	*виконано*
6	Написання та оформлення статті та тез доповіді	20.05 – 23.05.24	*виконано*
7	Підготовка пояснювальної записки	01.05 – 26.05.24	*виконано*
8	Підготовка презентації та доповіді	26.04 – 2.05.24	*виконано*
9	Нормоконтроль	03.06 – 08.06.24	*виконано*
10	Рецензування	08.06 – 14.06.24	*виконано*
11	Занесення диплома в електронний архів	15.06.2024	*виконано*
12	Попередній захист	15.06.2024	*виконано*
13	Допуск до захисту у зав. кафедри	18.06.2024	*виконано*

Дата видачі завдання «29» березня 2024 р.

Студент _____ Пархоменко Б. Є.

(підпис)

Керівник роботи_____ доц. кафедри ПІ Лановий О. Ф.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить 71 ст., 28 рис., 12 джерел.

АНАЛІЗ, ЕФЕКТИВНІСТЬ, МЕРЕЖІ ПЕТРІ, МОДЕЛЮВАННЯ, ПОШУК ФУНКЦІЙ, ПРОГРАМНА СИСТЕМА, ТЕОРІЯ АВТОМАТІВ, ТЕХНОЛОГІЧНІ АСПЕКТИ.

Об'єктом дослідження є процес розробки програмної системи, який включає в себе створення моделі, її аналіз та оптимізацію за допомогою методів теорії автоматів та мереж Петрі.

Головною метою роботи є вивчення та оцінка ефективності використання теорії автоматів та мереж Петрі у процесі розробки програмної системи. Зокрема, робота спрямована на аналіз можливостей абстрагування логічних взаємодій та моделювання паралельних процесів за допомогою цих теоретичних підходів.

Для досягнення поставленої мети використовується метод комплексного аналізу та моделювання. Застосовуються базові концепції теорії автоматів для моделювання логічних аспектів програмних систем, а також використання мереж Петрі для аналізу та оптимізації паралельних та розподілених програмних процесів.

Отримані результати вказують на те, що використання теорії автоматів та мереж Петрі у процесі розробки програмного забезпечення є дієвим та обґрунтованим. Моделювання логічних взаємодій за допомогою автоматів дозволяє ефективно абстрагувати складність системи, тоді як мережі Петрі сприяють аналізу та оптимізації паралельних процесів. Результати дослідження можуть бути використані для покращення процесів розробки, зменшення ризиків та підвищення якості програмного продукту.

ANALYSIS, EFFICIENCY, PETRI NETS, SIMULATION, FUNCTION SEARCH, SOFTWARE SYSTEM, AUTOMATA THEORY, TECHNOLOGICAL ASPECTS.

The object of research is the process of developing a software system, which includes the creation of a model, its analysis and optimization using the methods of automata theory and Petri nets.

The main goal of the course work is to study and evaluate the effectiveness of using the theory of automata and Petri nets in the process of developing a software system. In particular, the work is aimed at analyzing the possibilities of abstracting logical interactions and modeling parallel processes using these theoretical approaches.

To achieve the goal, the method of complex analysis and modeling is used. The basic concepts of automata theory are applied to model the logical aspects of software systems, as well as the use of Petri nets to analyze and optimize parallel and distributed software processes.

The obtained results indicate that the use of the theory of automata and Petri nets in the process of software development is effective and justified. Modeling logical interactions with the help of automata allows you to effectively abstract the complexity of the system, while Petri nets facilitate the analysis and optimization of parallel processes. Research results can be used to improve development processes, reduce risks, and improve software product quality.

Я, Пархоменко Богдан Євгенович, студент гр. ІПЗм-22-6, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження процесу розробки моделі програмної системи методами теорії автоматів та мереж Петрі», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

В сучасному інформаційному суспільстві розробка програмних систем визначається непередбачуваною динамікою вимог, технологічними інноваціями та постійно зростаючою складністю функціоналу. Щоб ефективно впоратися з цими викликами, розробники програмного забезпечення повинні використовувати передові методології та інструменти, які дозволяють систематизувати, аналізувати та оптимізувати процес розробки.

Одним із ключових напрямків у сфері розробки програмних систем є використання теорії автоматів та мереж Петрі для моделювання та аналізу їхніх структур і процесів. Теорія автоматів дозволяє абстрагувати складні логічні взаємодії та стани програмної системи, тоді як мережі Петрі надають засіб для моделювання та аналізу паралельних та розподілених процесів.

Ця робота спрямована на дослідження процесу розробки моделі програмної системи, зокрема використання методів теорії автоматів та мереж Петрі у цьому контексті. Вона має на меті висвітлити ефективність та практичність використання цих теоретичних підходів для покращення процесів проектування, аналізу та впровадження програмного забезпечення.

У цьому контексті важливо розглянути базові концепції теорії автоматів, їх застосування для моделювання логічних аспектів програмних систем, а також вивчити можливості мереж Петрі для опису та аналізу різноманітних програмних процесів.

Дослідження цих аспектів дозволить отримати глибше розуміння того, як використання теорії автоматів та мереж Петрі може сприяти оптимізації та удосконаленню процесу розробки програмного забезпечення.

1 ТЕОРЕТИЧНІ ОСНОВИ ТЕОРІЇ АВТОМАТІВ ТА МЕРЕЖ ПЕТРІ

1.1 Визначення теорії автоматів та її роль у розробці програмних систем

Теорія автоматів – це розділ теоретичної інформатики, що вивчає абстрактні машини (автомати) та проблеми, які можуть бути вирішені за допомогою цих машин. Вона включає в себе формальні моделі обчислень та їх застосування для моделювання систем, особливо програмного забезпечення (ПЗ).

Види автоматів:

а) скінченні автомати (Finite Automata):

1) детермінований скінченний автомат (DFA): кожен стан має один вихід для кожного можливого вхідного символу;
2) недетермінований скінченний автомат (NFA): кожен стан може мати кілька виходів для одного вхідного символу або не мати жодного;
3) автомати з кінцевою пам'яттю: вони можуть мати стеки або черги для зберігання додаткової інформації;

б) автомати з магазинною пам’яттю (Pushdown Automata, PDA):

1) використовують стек для зберігання додаткової інформації;
2) використовуються для аналізу контекстно-вільних мов;

в) лінійно-обмежені автомати (Linear Bounded Automata, LBA):

1) обмежені за кількістю пам'яті, яку вони можуть використовувати;
2) використовуються для аналізу контекстно-залежних мов;

г) тюрінгові машини (Turing Machines):

1) моделюють будь-яку обчислювану функцію;
2) мають необмежену стрічку пам'яті, що дозволяє їм виконувати будь-який алгоритм.

Теорія автоматів є галуззю теоретичного інформаційного наукоємного аналізу та вивчення автоматичних обчислювальних процесів та систем. Основна мета теорії автоматів полягає в розумінні та моделюванні логічних взаємодій в системах, що можуть бути представлені у вигляді автоматів [1].

В контексті розробки програмних систем теорія автоматів визначає набір концепцій та методів, які дозволяють абстрагувати та формалізувати процеси, що

відбуваються в межах програмного забезпечення. Вона надає можливість структурувати та виокремити ключові аспекти логіки функціонування системи, що полегшує розуміння та аналіз її роботи.

Роль теорії автоматів у розробці програмних систем полягає в наступних аспектах:

- моделювання поведінки системи: Теорія автоматів дозволяє концептуалізувати роботу програмного забезпечення через визначення станів, переходів між станами та обробки вхідних даних;
- управління складністю систем: Автомати надають ефективний спосіб розбити систему на менші, управляючі частини, що спрощує розробку та розуміння великих програмних проектів;
- тестування та верифікація: За допомогою автоматів можна формалізувати та провести тестування різних аспектів системи, а також використовувати їх для верифікації правильності програмного коду;
- оптимізація процесів розробки: Застосування теорії автоматів дозволяє зменшити ризики, пов'язані з розробкою програмного забезпечення, шляхом систематизації та формалізації робочих процесів.

Вивчення та використання теорії автоматів стає важливим елементом сучасної розробки програмного забезпечення, сприяючи покращенню ефективності та надійності програмних систем.

1.2 Основні поняття теорії автоматів

У цьому розділі розглядаються основні поняття теорії автоматів, які є ключовими для розуміння та моделювання логічних взаємодій у програмних системах за допомогою автоматів [2]:

а) стан:

1) визначення: стан в теорії автоматів є однією з можливих конфігурацій системи. це визначений момент часу, в якому система знаходиться під час свого функціонування;

2) роль: стани визначають можливі умови або ситуації, в яких може перебувати автомат;

б) перехід:

1) визначення: перехід – це зміна стану автомата, яка відбувається відповідно до визначених правил;
2) роль: переходи описують, як автомат реагує на вхідні дані та яким чином змінює свій стан;

в) алфавіт:

1) визначення: алфавіт представляє собою скінчений набір символів, які можуть бути використані як вхідні дані для автомата;
2) роль: алфавіт визначає множину символів, які автомат може сприймати та обробляти;

г) автомат:

1) визначення: автомат є математичною моделлю, що складається зі станів, переходів між станами та механізму обробки вхідних даних;
2) роль: автомати використовуються для моделювання та вивчення поведінки систем, де важливим є порядок подій та переходів між різними станами.

Ці поняття становлять основу для розуміння та аналізу автоматичних процесів та взаємодій у програмних системах, що є важливим елементом при використанні теорії автоматів у розробці програмного забезпечення.

1.3 Огляд теорії мереж Петрі та їх застосування у моделюванні процесів

Однією з ключових тем сучасної інформаційної технології є моделювання та оптимізація процесів. У цьому контексті теорія мереж Петрі займає важливе місце, надаючи засіб для глибокого аналізу та дослідження різних систем [3].

Мережі Петрі базуються на ідеї графічного представлення взаємодій між подіями та станами у системі. Основні елементи мереж Петрі – це позиції (стан системи), переходи (події) та дуги, які показують взаємозв'язки між ними. Це

дозволяє візуально моделювати та аналізувати процеси, спрощуючи сприйняття та розуміння логіки подій.

Однією з важливих особливостей мереж Петрі є їхній потенціал у моделюванні паралельних процесів. Вони ефективно вирішують завдання, пов'язані з управлінням різними подіями, які можуть відбуватися одночасно. Це робить їх популярним інструментом для представлення та аналізу складних систем, в яких важливо враховувати взаємозв'язки подій.

Мережі Петрі також широко застосовуються для моделювання систем масштабування. Вони дозволяють досліджувати, як зміни у розмірі та обсязі системи впливають на її функціонування та ефективність.

Застосування мереж Петрі у сучасній інженерії програмного забезпечення та системному аналізі розширюються на аналіз та оптимізацію ресурсів, виявлення конфліктів та розробку стратегій оптимального розвитку.

1.4 Наукові аспекти використання теорії автоматів для моделювання ПЗ

Теорія автоматів є основоположною в теоретичній інформатиці та має важливе значення для моделювання програмного забезпечення. Вона дозволяє використовувати математичні моделі для опису та аналізу обчислювальних процесів і систем. Серед найважливіших аспектів застосування теорії автоматів у цій сфері можна виділити формальні методи, верифікацію, аналіз продуктивності та автоматизацію процесів розробки.

Формальні методи, що базуються на теорії автоматів, дозволяють точно описувати поведінку програмних систем. Використовуючи математичний апарат, можна аналізувати та доводити властивості програм, такі як коректність, завершеність та безпечність. Наприклад, скінченні автомати широко використовуються для опису дискретних систем, де стан та переходи між станами визначаються чітко.

Верифікація і валідація програмного забезпечення є критичними для забезпечення його якості та надійності. За допомогою теорії автоматів моделюються системи у вигляді автоматів, що дозволяє застосовувати методи

перевірки моделей (model checking). Це особливо важливо для критичних систем, таких як авіаційні або медичні, де помилки можуть мати серйозні наслідки.

Теорія автоматів також відіграє важливу роль в аналізі продуктивності програмних систем. Вона дозволяє моделювати асинхронні та паралельні обчислення, допомагаючи виявити потенційні вузькі місця та проблеми з масштабованістю. Наприклад, скінченні автомати можуть моделювати системи з великою кількістю станів та переходів, що сприяє більш точному аналізу їхньої продуктивності.

Автоматизація процесів розробки є ще одним важливим аспектом застосування теорії автоматів. Вона дозволяє автоматизувати різні етапи розробки програмного забезпечення, такі як генерація коду, тестування та синтез програм. На основі специфікацій, представлених у вигляді автоматів, можуть автоматично генеруватися тести, що перевіряють всі можливі стани та переходи системи.

Крім того, теорія автоматів використовується для моделювання користувацьких інтерфейсів та інтерактивних систем. Це забезпечує передбачуваність та протестованість всіх можливих дій користувача та реакцій системи. Автомати також застосовуються для моделювання комунікаційних протоколів, що дозволяє перевірити їх коректність та ефективність, описуючи стан кожного учасника комунікації та їхні дії залежно від отриманих повідомлень.

Також теорія автоматів тісно пов’язана з формальними мовами та розробкою компіляторів. Контекстно-вільні граматики, що є частиною теорії автоматів, використовуються для опису синтаксису програмних мов, а автомати застосовуються для аналізу та синтаксичного розбору коду.

Таким чином, теорія автоматів забезпечує формальний, строгий та ефективний підхід до проектування, аналізу та верифікації програмного забезпечення, що сприяє підвищенню його якості, надійності та оптимізації процесу розробки.

1.5 Моделювання програмного забезпечення

Моделювання програмного забезпечення (ПЗ) відіграє критично важливу роль у процесі його розробки. Основна мета моделювання полягає в тому, щоб створити абстрактне уявлення про систему, яке допомагає краще зрозуміти її структуру та поведінку. Це особливо корисно на ранніх етапах, коли потрібно визначити основні вимоги та розробити архітектуру.

Однією з ключових переваг моделювання є полегшення комунікації між усіма учасниками проекту. Моделі служать спільною мовою для розробників, аналітиків, менеджерів та інших зацікавлених сторін. Це дозволяє їм ефективніше обговорювати вимоги та рішення, зменшуючи ризик непорозумінь і помилок. Моделі допомагають також перевірити вимоги до ПЗ, виявити їхню неповноту, суперечності або надмірність, що сприяє більш точному визначенню обсягів робіт.

Процес моделювання також сприяє проектуванню системи, допомагаючи визначити взаємодію між компонентами та обрати оптимальні технології. Це дозволяє створити більш ефективну та надійну архітектуру. Моделювання дає можливість проводити верифікацію та валідацію системи ще до її фактичної реалізації, що допомагає виявити та виправити помилки на ранніх етапах, значно знижуючи витрати на їхнє виправлення в майбутньому.

Крім того, моделювання дає змогу оцінити продуктивність системи та її компонентів, що допомагає виявити потенційні вузькі місця та оптимізувати використання ресурсів. Це забезпечує необхідний рівень продуктивності та стабільності системи.

Моделі можуть також використовуватися для автоматичної генерації тестових сценаріїв, що підвищує якість тестування та зменшує час, необхідний для ручного тестування. Моделювання користувацького інтерфейсу допомагає оцінити зручність використання системи, забезпечуючи позитивний користувацький досвід.

На завершення, моделювання програмного забезпечення є важливим інструментом для планування та управління проектами. Воно допомагає точніше

визначити обсяги робіт, часові рамки та необхідні ресурси, що сприяє кращому контролю за виконанням проекту та зниженню ризиків. Таким чином, моделювання забезпечує більш структурований, ефективний та якісний підхід до розробки програмного забезпечення.

1.6 Постановка задачі

Розробити методику використання теорії автоматів та мереж Петрі для моделювання та аналізу алгоритмів, обґрунтувати вибір інструментів моделювання та створити програмну систему для побудови моделей алгоритмів:

а) дослідити теоретичні основи теорії автоматів та мереж Петрі, їх роль у розробці програмних систем:
   1) визначення теорії автоматів та її роль у розробці програмних систем;
   2) вивчення основних понять теорії автоматів;
   3) огляд теорії мереж Петрі та їх застосування у моделюванні процесів;

б) обґрунтувати вибір інструментів моделювання:
   1) визначити сферу використання теорії мереж Петрі;
   2) розглянути мережі Петрі як інструмент для створення моделей, включаючи їх розширення та використання кольорів, розв'язання конфліктів та концепцію під моделі;
   3) формалізувати алгоритм за допомогою мереж цифрових автоматів;

в) використовувати мережі Петрі для створення моделей алгоритмів:
   1) дослідити алгоритми моделювання;
   2) розробити модель мережі Петрі для алгоритму пошуку функцій;
   3) дослідити модель алгоритму пошуку функцій, включаючи особливості мереж Петрі, обмеження та забезпечення безпеки, збереження, активність та дерево досяжності;

г) описати програмну систему для побудови дерева функцій:
   1) визначити мету програмної системи;
   2) визначити функціональні характеристики, які потрібно врахувати;
   3) організувати програмний продукт;

2 ОБҐРУНТУВАННЯ ВИБОРУ ІНСТРУМЕНТІВ МОДЕЛЮВАННЯ

2.1 Сфера використання теорії мереж Петрі

Мережі Петрі, як інструмент для дослідження систем, відкривають можливість математичного моделювання систем у вигляді мереж Петрі. Теорія мереж Петрі передбачає, що аналіз таких мереж надають важливу інформацію про структуру та динамічну поведінку модельованої системи [4]. Ця інформація корисна для оцінки системи та розробки пропозицій щодо її удосконалення.

На сьогоднішній день мережі Петрі широко використовуються в галузі моделювання. В багатьох наукових областях вивчення явищ проводиться не безпосередньо, а через створення моделей. Модель – це уявлення, зазвичай в математичних термінах, того, що вважається найбільш характерним для досліджуваного об'єкта або системи. Маніпулюючи моделлю системи, можна отримати нові знання про неї, уникаючи складнощі та вартості аналізу самої реальної системи.

Зазвичай моделі мають математичну основу. Багато фізичних явищ можна описати числами, і їх зв'язок виражається рівняннями та нерівностями. Зокрема, в природничих науках і техніці рівняннями описуються такі характеристики, як маса, положення в просторі, момент, прискорення і сили. Для успішного використання моделювання необхідне знання як модельованих явищ, так і властивостей методу моделювання.

З виникненням суперкомп'ютерів значно зросла ефективність та корисність моделювання. Представлення системи математичною моделлю і подальше перетворення цієї моделі в команди для процесора дозволило моделювання великих і більш складних систем, ніж коли-небудь. Це призвело до значних досліджень методів моделювання на комп'ютерах і самих комп'ютерів, які взяли участь в моделюванні як обчислювальні засоби та як об'єкти моделювання.

Обчислювальні системи (ОС) представляють собою великі та складні структури, що включають безліч взаємодіючих компонент. Кожна з цих компонент також може бути високоградаційної, оскільки вона взаємодіє з іншими

елементами системи. Мережі Петрі, зокрема, можуть використовуватися для моделювання потоку інформації або інших ресурсів у системі.

Це принципово важливо для різноманітних систем, таких як економічні, юридичні, системи управління дорожнім рухом чи хімічні системи, які складаються з багатьох окремих компонентів, що взаємодіють між собою складним чином. Незважаючи на різноманіття модельованих систем, існують спільні риси, які повинні бути враховані при створенні моделей. Одна з основних ідей полягає в тому, що системи складаються з окремих взаємодіючих компонентів, кожен з яких може бути системою сам по собі.

Стан кожної компоненти є абстракцією відповідної інформації, необхідної для опису її майбутніх дій. Оскільки стан компоненти часто залежить від її передісторії, він може змінюватися з часом.

Зі специфічною природою дій в системі пов'язані певні труднощі в моделюванні. Оскільки компоненти системи взаємодіють, важливо встановлювати синхронізацію. Пересилання інформації чи матеріалів від однієї компоненти до іншої вимагає, щоб дії, пов'язані з цим обміном, були вчасно синхронізовані. Це може призвести до ситуації, коли одна компонента чекає на іншу. Узгодження в часі дій різних компонент може бути вкрай складним, особливо при описі отриманих в результаті взаємодії між компонентами складних взаємодій.

Моделювання само по собі є обмеженим у своїй корисності, важливо провести аналіз модельованої системи для глибокого розуміння її поведінки. Розвиток теорії мереж Петрі розглядається у двох основних напрямках.

Формальна теорія мереж Петрі займається розробкою основних засобів, методів і понять, необхідних для застосування мереж Петрі. Прикладна теорія мереж Петрі, з іншого боку, пов'язана головним чином із застосуванням мереж Петрі для моделювання систем, їхнього аналізу та глибокого проникнення в модельовану систему.

Існують кілька шляхів практичного застосування мереж Петрі при проектуванні та аналізі систем. В одному підході мережі Петрі розглядаються як

інструмент для допоміжного аналізу. Тут для побудови системи використовуються стандартні методи проектування, після чого система моделюється мережею Петрі, а модель аналізується. Будь-які труднощі, виявлені під час аналізу, вказують на недоліки в проекті, які потребують коригування. Цей цикл повторюється до тих пір, поки аналіз не призведе до успішних результатів. Інший підхід передбачає використання методів аналізу лише для створення безпомилкового проекту мережі Петрі, який потім трансформується в реальну робочу систему.

Інтерпретація мереж Петрі ґрунтується на концепціях умов та подій. Стан системи визначається сукупністю умов, де умова представляє собою предикат або логічний опис стану системи, приймаючи значення «істина » або «брехня». Робота системи полягає в виконанні послідовності подій.

Оскільки події представляють собою дії, вони можуть відбуватися. Для виникнення події необхідно виконання певних умов, які називаються передумовами. Виникнення подій може призвести до порушення передумов і може спричинити виконання умов, які називаються постумовою. У мережі Петрі умови моделюються позиціями, а події – переходами. Передумови події представлені вхідними позиціями відповідного переходу, постумови – вихідними позиціями.

Моделювання в мережах Петрі здійснюється на подієвому рівні, визначаючи, які дії відбуваються в системі, які стани передували цим діям і які стани система приймає після виконання дії. Виконання подієвої моделі в мережах Петрі описує поведінку системи. Аналіз результатів виконання може вказати на те, в яких станах система перебувала або не пробувала, які стани в принципі недосяжні. Однак такий аналіз не надає числових характеристик, що визначають стан системи.

Розвиток теорії мереж Петрі привів до виникнення «кольорових» мереж Петрі, які використовуються для моделювання змінних, типів даних, умов та інших конструкцій, близьких до мов програмування. Вони є більш адаптованими до програмного середовища.

Мережі Петрі можуть також успішно моделювати апаратне забезпечення ЕОМ на різних рівнях, від простих блоків до великих обчислювальних систем. Вони використовуються для моделювання і аналізу як апаратного, так і програмного забезпечення. Мережі Петрі дозволяють визначати стани системи та її поведінку, а також ефективно використовуватися для практичного застосування в області опису та моделювання програмного забезпечення.

Більшість зусиль у сфері застосування мереж Петрі пов'язана з аналізом, специфікацією і описом послідовних програм, однак системи паралельних процесів залишаються важливою дослідницькою проблемою. Навіть з усіма перевагами, описаними вище, мережі Петрі мають обмеження в ролі мови програмування у виконанні в обчислювальній системі. Вони не мають чіткого поняття процесу, який можна виконати на конкретному процесорі, і відсутній чіткий порядок виконання, оскільки теорія виходить з опису паралельних процесів.

Мережі Петрі мають високий потенціал для опису паралельних систем, який не менш потужний, ніж PVM, MPI, SDL і інші. Проте для їх виконання на процесорах необхідно внести зміни в опис паралельного розподілення.

Моделювання системи передбачає декомпозицію, щоб виявити всі структурні елементи системи залежно від рівня деталізації. Компоненти системи і їх дії можна уявити як абстрактні події, які можуть відбуватися один раз, повторюватися багаторазово, породжуючи конкретні дії, або не відбуватися взагалі. Сукупність дій, що виникають як реалізації подій, утворює процес. Узагальнюючи, та ж сама система може функціонувати в одних і тих самих умовах по-різному, породжуючи безліч різних процесів.

Справжня система функціонує в часі, і події відбуваються в конкретні моменти часу, триваючи протягом певного періоду. Однак строге врахування часу при моделюванні великих паралельних систем може призвести до деяких недоліків:

Громіздкість моделі: Велика система вимагає врахування стану всіх її компонентів при кожній зміні загального стану, що може зробити модель дуже складною і важкоуправній.

Втрата причинно-наслідкових зв'язків: Строго визначений час може викликати непевність у відношенні до причинно-наслідкових зв'язків між подіями в системі. Якщо дві події відбулися одночасно, важко визначити, чи було це випадково чи намірено.

Складність вираження конфліктів і очікувань: Поняття конфліктів між компонентами системи або очікувань результатів роботи одного компонента від іншого можуть бути важко виражені в термінах змін стану системи, які залежать від часу.

Невизначеність часових інтервалів: Події можуть відбуватися в системі протягом невизначено великих інтервалів часу, і точний час їх початку, кінця та тривалість може бути важко або неможливо визначити наперед.

Виходом з цього може бути відмова від введення дискретних систем часу та тактованих послідовностей змін станів. Замість цього можна використовувати причинно-наслідкові зв'язки між подіями, що добре виражаються за допомогою мереж Петрі.

2.2 Мережі Петрі, як інструмент для створення моделей

Інтерпретація мереж Петрі базується на концепціях умов і подій. Стан системи характеризується сукупністю умов. Робота системи передбачає послідовність подій. Для виникнення події потрібно виконання певних умов, які називаються передумовами. Виникнення події може призвести до виконання умов, які називаються постумовами. У мережі Петрі умови моделюються позиціями, а події – переходами. Передумови події виражаються вхідними позиціями відповідного переходу, а постумови – вихідними позиціями.

Оскільки мережі Петрі моделюють події як миттєві та асинхронні, а їх взаємозв'язок надає асинхронний характер, цей підхід є зручним для представлення різних взаємозалежних та паралельних процесів. Застосування

мереж Петрі в завданнях, пов'язаних з розподілом ресурсів, привабливе через його очевидність, адекватність та технологічні переваги при реалізації моделей на ЕОМ. Мережа Петрі характеризується набором (формула 2.1):

$$PN = < S, T, F, W, M_0 >, \quad (2.1)$$

де S = {s1, s2,..., sm} – нескінченний набір позицій;

T = {t1, t2, ..., tn} – нескінченний набір переходів;

F $\subseteq$ SxT $\cup$ TxS – необмежена кількість орієнтованих зв'язків (відношення інцидентності);

W: F $\rightarrow$ N – функція кратності дуг;

M_0: P $\rightarrow$ N – початкова розмітка визначає умови, при яких можуть відбутися переходи.

Мережа Петрі представлена позиціями та переходами, які взаємодіють за допомогою орієнтованих дуг, що передають мітки. Вага дуги визначає кількість міток, які вилучаються або розміщуються в позиції. Мітки існують тільки в позиціях, які інтерпретують стан системи, і кількість міток визначається як маркування (див. рисунок 2.1).

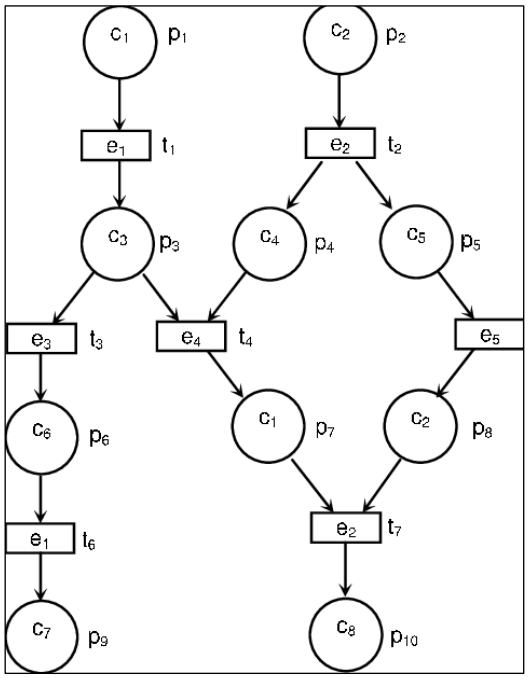


Рисунок 2.1 – Мічена сітка Петрі (маркована сітка, за даними [5])

Передумова показує, що кількість міток у вхідній позиції повинна бути не менше ваги дуги, яка з'єднує цю позицію і перехід. Постумова вказує на те, що вихідна позиція може забрати кількість міток, яка повинна бути не менше ваги дуги, що з'єднує перехід і цю позицію. Тільки коли всі передумови і постумови виконані, перехід може спрацювати, тобто мітки з вхідних позицій переміщаються в вихідні позиції, викликаючи подію, яка визначає зміну стану системи.

2.2.1 Розширення мережі Петрі

Для використання мереж Петрі в моделюванні випадкових процесів були впроваджені такі розширення: використання часу (стохастичні мережі Петрі) [6]. Моделювання різноманітних процесів вимагає кількісного розгляду часу, для чого стандарт мереж Петрі не надає відповідних механізмів. Стохастичні мережі Петрі, як розширення стандарту, здатні ураховувати час. Загалом, будь-які компоненти мережі Петрі можна поєднати з компонентами часу. Так, час може бути врахований в позиціях, мітках, дугах або переходах. Дії, пов'язані зі споживанням часу, описуються переходами, оскільки стани системи визначаються позиціями. Якщо мітки деяких позицій розглядаються як послуги, що виконуються за фіксований час, то такі позиції можуть бути замінені переходом. Крім того, мітки відповідних переходів не затримуються в них протягом часу роботи, але встановлюються як недоступні (зарезервовані) у вихідних позиціях. Це допомагає уникнути ситуації, коли паралельні процеси можуть перешкоджати спрацьовуванню переходу протягом часу надання послуги. Такий підхід вимагає повернення міток на вхідні позиції, що відображає реальні процеси і не вписується в концепцію односпрямованого потоку, яка використовується в стандарті мереж Петрі. Оскільки для опису реальних процесів необхідно враховувати не лише фіксований час, а й час, що описується статистичними розподілами, то для таких випадків використовуються стохастичні мережі Петрі як особливий вид часового оцінювання функціонування мережі Петрі. Коли перехід отримує дозвіл на спрацювання, він виконується негайно, і мітки з

вхідних позицій переміщуються в вихідні позиції. Час послуги може бути описаний одним із наступних розподілів:

- розподіл з постійною інтенсивністю;
- розподіл, який має однакову інтенсивність на кожному відрізку;
- розподіл із експоненційною щільністю ймовірності;
- розподіл, що визначається сумою k незалежних випадкових подій з експоненційним розподілом;
- розподіл, який описує кількість незалежних інцидентів в одиничний період часу;
- розподіл із симетричною кривою щільності ймовірності;
- розподіл зі схожістю до експоненційного, але зі зміненим параметром форми;
- розподіл із двома параметрами, широко використовується для моделювання випадкових величин;
- розподіл із двома параметрами, який може моделювати різні типи випадкових величин;
- розподіл, який описується трикутною функцією щільності ймовірності і відомий своєю симетричністю.

2.2.2 Мережі Петрі з використанням кольорів

Мережі Петрі з використанням кольорів є розширенням стандартних мереж Петрі, яке дозволяє призначати різні кольори (мітки) об'єктам та ресурсам в системі [7]. Це розширення надає можливість більш точного та гнучкого моделювання процесів, де кожна мітка може мати свої характеристики.

Основна ідея полягає в тому, що замість використання одного виду міток для всіх об'єктів, можна використовувати різні кольори міток для різних типів об'єктів або для різних станів об'єктів у системі. Це дозволяє моделювати багатокритеріальні системи та різноманітні умови їх функціонування.

З використанням кольорів можна визначити різні параметри, такі як пріоритети, ресурси чи стани, які мають вплив на хід подій у системі. Мережі

Петрі з кольорами дозволяють виразити складні взаємодії та умови, що часто виникають в реальних системах.

2.2.3 Розв'язання конфліктів

Розв'язання конфліктів у контексті мереж Петрі включає в себе вирішення ситуацій, де кілька переходів одночасно конкурують за ресурси або викликають взаємну блокаду [8]. Для вирішення таких ситуацій можна використовувати різні стратегії:

а) пріоритети переходів:

1) визначення пріоритетів для різних переходів дозволяє встановити порядок їх виконання;
2) можливість задавати вагу або пріоритет кожному переходу, щоб система віддавала перевагу виконанню певних дій;

б) буферизація:

1) введення проміжних позицій або буферів дозволяє тимчасово зберігати мітки, уникнути конфліктів та забезпечити синхронізацію між переходами;
2) система може зберігати мітки в буфері до тих пір, поки не виконаються певні умови для їхнього переміщення;

в) синхронізація:

1) використання синхронізуючих механізмів, таких як семафори або блокування, дозволяє керувати доступом до ресурсів та запобігати конфліктам;
2) застосування умов, які пов'язані із станом ресурсів або позицій, може визначати, коли дозволяти виконання переходів;

г) розподіл ресурсів:

1) створення додаткових позицій та дуг для розподілу обмежених ресурсів допомагає управляти доступом переходів до цих ресурсів;
2) система визначає, які переходи можуть отримати доступ до ресурсів у певний час;

д) механізми блокування:

1) застосування блокування або інших механізмів очікування дозволяє визначити умови виконання переходів;
2) можливість встановлювати блокування на певному стані системи або наявності міток у позиціях.

Ці стратегії можуть бути комбіновані для досягнення оптимальних результатів в конкретних сценаріях моделювання систем за допомогою Мереж Петрі. Вибір конкретної стратегії залежить від особливостей системи та поставлених завдань.

2.2.4 Концепція підмоделі

Підмодель в контексті Мереж Петрі вказує на частину або окремий аспект системи, який може бути взятий окремо для аналізу чи моделювання. Введення поняття підмоделі дозволяє розділити велику та складну систему на менші, більш керовані компоненти для полегшення аналізу та розуміння.

Основні аспекти підмоделі в Мережах Петрі можуть включати:

- підмодельні позиції та переходи: визначення конкретних позицій та переходів, які становлять підмодель, дозволяє сконцентруватися на певних аспектах системи;
- взаємодія: встановлення правил взаємодії між підмоделями. Це може включати в себе обмін сигналами, обмін ресурсами або інші форми взаємодії;
- орієнтованість на задачу: визначення, яку конкретну задачу вирішує підмодель. Це може бути аналіз певного аспекту системи або взаємодія конкретних компонентів;
- рівень абстракції: визначення рівня абстракції, на якому розглядається підмодель. Вона може бути вкрай деталізованою або, навпаки, зосередженою на високорівневих аспектах;
- аналіз та моделювання: підмодель може бути створена для використання в аналізі, моделюванні або вирішенні конкретних завдань.

Підмоделі в мережах Петрі дозволяють розбити складні системи на менші, керовані частини, що полегшує розуміння та аналіз системи в цілому. Окремі підмоделі можуть використовуватися для дослідження конкретних аспектів, і їх взаємодія визначає поведінку системи в цілому.

2.3 Формалізація алгоритму за допомогою мереж цифрових автоматів

Загальний алгоритм роботи методу передачі кадрів STTE можна представити за допомогою цифрових автоматів (ЦА). У кольорових мережах Петрі колір маркера використовується у функціях передачі маркера по дугах, що дозволяє зменшити кількість переходів та позицій шляхом призначення дугам або переходам (у сегменті коду переходу) функцій передачі маркера. Використання кольорових мереж Петрі дозволяє оптимізувати структуру мережі. ЦА може розглядатися як окремий випадок мереж Петрі.

Існують такі способи опису цифрового автомата: граф-схема алгоритму (ГСА), система канонічних рівнянь та вихідних функцій, а також таблиця переходів станів. Мережа Петрі визначається кортежем і відрізняється від цифрового автомата. У пакеті CPN Tool кортеж складається з дев'яти елементів CPN (P,T,A,Σ,V,C,G,E,I), де:

- P – остаточна множина позицій;
- T – остаточна множина переходів;
- A – $A \subseteq P \times T \cup T \times P$ безліч спрямованих дуг;
- $\sum$ – не порожня кінцева множина кольорів; V – кінцева множина типів змінних;
- $C:P \rightarrow \Sigma$ – функція множини кольору;
- $G:T \rightarrow EXPRV$ – функція умови спрацьовування переходу;
- $E:A \rightarrow EXPRV$ – функція опису дуги;
- $I:P \rightarrow EXPR0$ – функція ініціалізації.

EXPR представляє множину виразів, записаних мовою CPN ML, а Type[e] вказує на тип виразу $e \in EXPR$, тобто тип значень, отриманих при обчисленні

виразу e. Множина вільних змінних у виразі e позначається як Var[e], а тип змінної v позначається як Type[v]. Вільна змінна – це змінна, яка не пов'язана з локальним середовищем виразу. Var[e] позначається як EXPRV. Вираз ініціалізації I має бути замкненим, тобто не повинен містити вільних змінних, тому EXPR0.

У кольорових мережах Петрі колір маркера використовується у функціях передачі маркера по дугах мережі Петрі, що дозволяє зменшити кількість переходів і позицій завдяки призначенню дугам або переходам (у сегменті коду переходу) функцій передачі маркера. ЦА може бути окремим випадком побудови мереж Петрі, тому можна легко створити імітаційну модель на основі мереж Петрі з використанням ЦА.

Цільова аудиторія представлена у вигляді орієнтованого графа, який складається зі станів автомата (вершин) та вхідних сигналів. Існують детерміновані автомати Мілі та Мура. Особливістю автоматів Мура є те, що вихідний сигнал залежить не від вхідного сигналу, а лише від стану переходу. Розглянемо орієнтований граф автомата Мура з STTE, який зображено на рисунку 2.2. Із стану S0 при надходженні одиничного вхідного сигналу автомат переходить у стан S1. Із стану S1 при одиничному вхідному сигналі він переходить у стан S2. Із стану S2 при сигналі X1&X2&X3&X4 автомат переходить у стан S3, при сигналі X1&¬X2&¬X3&X4 – у стан S8, при сигналі X1&X2&X3 – у стан S4, при сигналі ¬X1 – у стан S6, а при сигналі X1&X2 – у стан S5.

За графом ЦА Мура складемо таблицю переходів автомата (рисунок 2.3) під дією вхідних сигналів. У першій колонці Si(t) зазначений стан №i автомата в поточний момент часу t, у другій колонці Sj(t+1) вказується стан №j автомата в наступний момент часу t, у третій колонці Xi,j(t) – вхідний сигнал у поточний час t, а в четвертій колонці Yj(t+1) – вихідний сигнал у наступний час t.

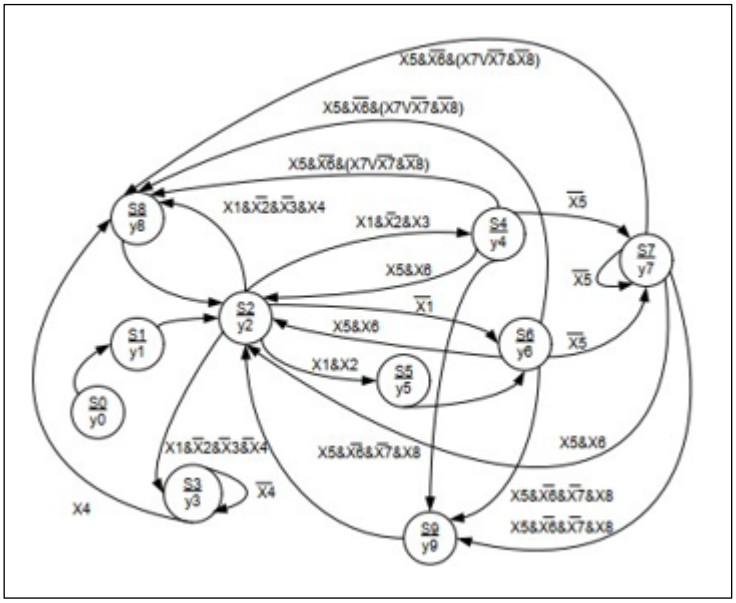


Рисунок 2.2 – Розробка алгоритмів за допомогою цифрових автоматів

У таблиці переходів послідовно перелічуються всі можливі переходи, починаючи з нульового стану до останнього (див. рис. 2.3). І так далі.

$S_i(t)$	$X_{i,j}(t)$	$S_j(t+1)/Y_j(t+1)$
S0	1	S1/y1
S1	1	S2/y2
S2	X1&¬X2&¬X3&¬X4	S3/y3
	X1&¬X2&X3	S4/y4
	X1&X2	S5/y5
	¬X1	S6/y6
	X1&¬X2&¬X3&X4	S8/y8
S3	¬X4	S3/y3
	X4	S8/y8
S4	X5&X6	S2/y2
	¬X5	S7/y7
	X5&¬X6&(X7 ∨ ¬X7&¬X8)	S8/y8
	X5&¬X6&¬X7&X8	S9/y9
S5	1	S6/y6
S6	X5&X6	S2/y2
	¬X5	S7/y7
	X5&¬X6&(X7 ∨ ¬X7&¬X8)	S8/y8
	X5&¬X6&¬X7&X8	S9/y9
S7	X5&X6	S2/y2
	¬X5	S7/y7
	X5&¬X6&(X7 ∨ ¬X7&¬X8)	S8/y8
	X5&¬X6&¬X7&X8	S9/y9
S8	1	S2/y2
S9	1	S2/y2

Рисунок 2.3 – Таблиця переходів ЦА

Інші компоненти ЦА також будуються за аналогічним методом шляхом трансформації блок-схем алгоритмів. Запропонований спосіб конвертації цифрових автоматів у мережі Петрі використовувався у дослідженнях для перетворення формалізації від ЦА до мереж Петрі. На рисунку 2.4 зображена підмережа системного захисника, яка була створена за допомогою описаного вище цифрового автомата та методики перетворення ЦА на мережу Петрі.

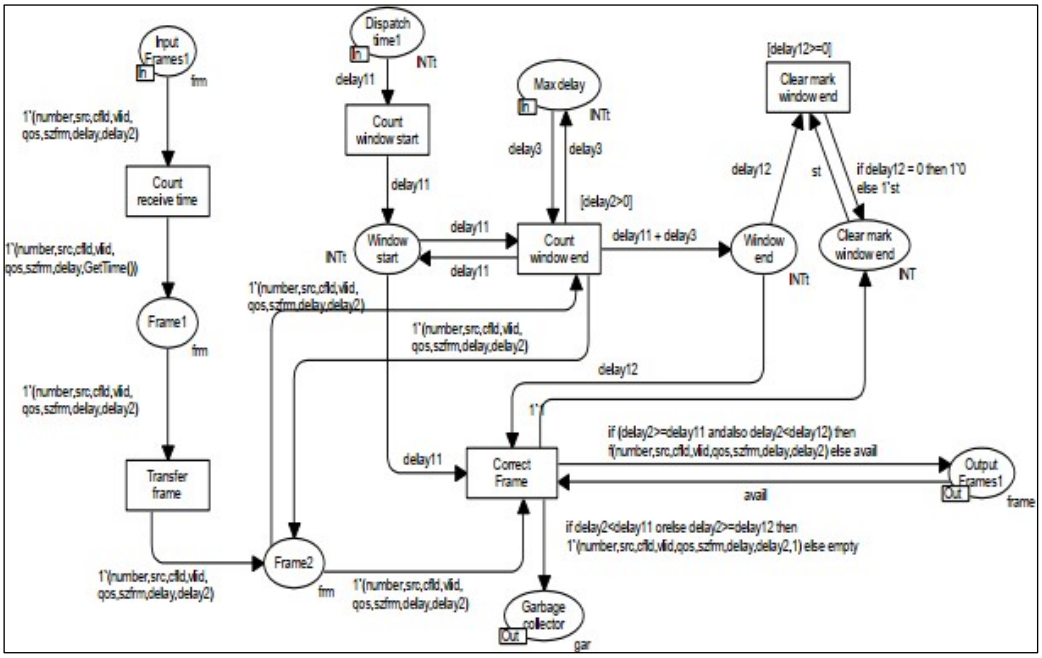


Рисунок 2.4 – Підмережа захисника системи

Фази S3 та S4 відповідають за початок і завершення тимчасового вікна захисника; стани S1 та S2 вказують на прийнятий захисником кадр; стан S6 визначає коректність позиції у мережі Петрі прийнятого кадру з урахуванням тимчасових обмежень; фази S7 та S8 – реакція на спрацювання тимчасових умов.

Вибір мереж Петрі як основного інструменту моделювання обґрунтований їхньою здатністю до точного відображення складних процесів та взаємодій у системі, забезпечуючи гнучкість та широкі аналітичні можливості. Цей інструмент є незамінним при розробці, оцінці та вдосконаленні систем, які включають складні взаємодії та процеси, забезпечуючи можливість ефективного моделювання різноманітних сценаріїв і взаємодій.

3 ВИКОРИСТАННЯ МЕРЕЖ ПЕТРІ ДЛЯ СТВОРЕННЯ МОДЕЛЕЙ АЛГОРИТМІВ

3.1 Алгоритми моделювання

Алгоритм – це чіткий і обмежений опис загального методу, заснованого на застосуванні конкретних елементарних тактів обробки. Загалом, під алгоритмом розуміється будь-яка точна інструкція, що задає обчислювальний процес (відомий як алгоритмічний), що розпочинається з довільного вихідного даного (з певного набору можливих вихідних даних) і спрямований на отримання повністю визначеного цим вихідним даним результату.

Поняття алгоритму займає центральне місце в сучасній математиці, зокрема в галузі обчислювальної математики. Наприклад, розв'язання числових рівнянь вимагає знаходження алгоритму, який, для будь-якої пари, складеної з рівняння цього типу та раціонального числа e, перетворює її в число (або набір чисел), менше, ніж e, що відрізняється від кореня (коренів) цього рівняння. Покращення обчислювальних машин дозволяє впроваджувати все більш складні алгоритми. Взагалі, вихідними даними та результатами алгоритму можуть бути найрізноманітніші конструктивні об'єкти, наприклад, результатами алгоритмів розпізнавання є слова «так» і «ні».

Робота алгоритму розпочинається підготовчим етапом, на якому можливе вихідне дане перетворюється в початковий член ряду, змінюючи один одного проміжні результати; це перетворення здійснюється за допомогою спеціального «правила початку», яке входить до складу даного алгоритму. Це правило полягає в застосуванні тотожного перетворення. Потім використовується «правило безпосередньої переробки», яке проводить послідовні перетворення кожного виникаючого проміжного результату в наступний. Ці перетворення продовжуються до тих пір, поки випробування, якому піддаються всі проміжні результати по мірі їх виникнення, не вказує, що даний проміжний результат є заключним; це випробування проводиться на основі спеціального «правила закінчення».

Якщо жоден з виникаючих проміжних результатів не спричиняє зупинку згідно з правилом закінчення, то або можна застосувати правило безпосередньої переробки до кожного з них, і алгоритмічний процес триватиме необмежено, або, якщо для якогось проміжного результату правило безпосередньої переробки виявиться несприйнятливим, процес завершиться без результату. У кінцевому підсумку, з заключного проміжного результату, також на підставі спеціального правила, виводиться остаточний результат. В багатьох важливих випадках правило початку і правило виведення результату визначають тотожні перетворення і тому окремо не формулюються. Таким чином, для характеризації кожного алгоритму можна виділити 7 параметрів:

- сукупність можливих вихідних даних;
- сукупність можливих результатів;
- сукупність проміжних результатів;
- правило початку;
- правило безпосередньої переробки;
- правило закінчення;
- правило виведення результату.

Комп'ютер – це обчислювальний пристрій, який не розуміє програму, а лише виконує її. Найбільш природний спосіб передачі інструкцій комп'ютеру – це запис їх у вигляді алгоритму на алгоритмічній мові. Сучасне значення слова «алгоритм» багато в чому аналогічне таким поняттям, як рецепт, процес, метод чи спосіб. Алгоритм володіє п'ятьма важливими властивостями:

Завершеність: алгоритм завжди має завершуватися після виконання кінцевої кількості кроків:

- визначеність: кожен крок алгоритму повинен бути чітко визначений;
- вхідні дані: алгоритм має вхідні дані, які задаються перед початком його виконання або визначаються динамічно під час його виконання;
- вихідні дані: алгоритм повинен мати одне або кілька вихідних даних, пов'язаних із вхідними даними;

– ефективність: алгоритм вважається ефективним, якщо його оператори достатньо прості для виконання за обмежений час за допомогою олівця і паперу.

Проблема створення алгоритму, який володіє конкретними властивостями, називається алгоритмічною проблемою. Зазвичай властивість шуканого алгоритму формулюється в термінах відповідності між вихідними даними і результатами алгоритму. Деякі важливі приклади алгоритмічних проблем включають проблему обчислення певної функції, дозволу даної множини та перерахування даної множини.

Нерозв'язність алгоритмічної проблеми вказує на відсутність відповідного алгоритму для її вирішення. Теореми, які встановлюють нерозв'язність таких проблем, вважаються серед найважливіших у теорії обчислень.

Області досліджень, пов'язані з алгоритмами, включають:

– аналіз алгоритмів: визначення робочих характеристик заданого алгоритму для визначення його ефективності, наприклад, швидкості виконання;
– теорія алгоритмів: вивчення існування та властивостей ефективних алгоритмів для обчислення певних величин;
– побудова алгоритмів: вивчення стандартних методів та прийомів, які використовуються при написанні алгоритмів.

Більшість практичних алгоритмів, що використовуються програмістами, входять в клас поліноміальних, що означає, що їх час роботи залежить не більше ніж від поліноміальної функції входу. Однак не всі задачі можна вирішити за поліноміальний час, а деякі задачі взагалі можуть бути невирішені жодним алгоритмом.

NP-повні задачі становлять особливий клас завдань, для яких невідомі поліноміальні алгоритми вирішення, і хоча не доведено, що такі алгоритми не існують, вони привертають увагу програмістів з декількох причин. Якщо вдалося довести, що певна задача є NP-повною, це може свідчити про її практичну нерозв'язність. У такому випадку розуміння цього дозволяє програмістам

витрачати час на розробку наближених алгоритмів замість безрезультатних спроб знаходження швидкого алгоритму для точного розв'язання.

Щодо прискорення виконання програм, основний приріст швидкості часто досягається не врахуванням особливостей системи, а алгоритмічною оптимізацією. Математичні методи, такі як біноміальні тотожності, рекурентні співвідношення, операторні методи і асимптотичні подання, широко використовуються для аналізу алгоритмів.

Теорія мереж Петрі, запропонована Петрі, розширила можливості аналізу алгоритмів. Вона стала ефективним методом для проектування, дослідження і моделювання систем у різних галузях, таких як економіка, фізика, обчислювальна техніка, біологія, астрономія та інші. Використання мереж Петрі дозволяє наочно представити всі події і процеси в системі, а також формально записати саму мережу мовою мереж Петрі.

Застосування теорії мереж Петрі до обчислювальних систем великим чином розширило розуміння про аналіз та синтез, але, на жаль це показалося досить точним, але не в повній мірі задовільнюючим. Існують описи основних методів моделювання систем за допомогою мереж Петрі, проте їх аналізу приділено дуже мало уваги.

Використання мереж Петрі для аналізу алгоритмів майже не розкрите, і, відповідно, проблема оптимізації алгоритмів за допомогою цієї теорії залишається практично невивченою. Таким чином, існує потреба у детальному вивченні теорії мереж Петрі в контексті алгоритмів.

3.2 Алгоритм пошуку функцій

Алгоритм аналізу функцій в програмному коді представляє собою семантичний аналізатор конструкцій. Вхідні дані для алгоритму – це вихідний код програми на мові «C» або «C++», який подається у вигляді набору символів; аналіз виконується посимвольно. Вихідними даними є дві таблиці функцій, які містять інформацію про знайдені функції і їх відносини. Алгоритм шукає лише ті функції, які оголошені програмістом, і ігнорує стандартні процедури.

Процес аналізу вихідного коду розпочинається перевіркою наявності вхідних даних, тобто перевіряється, чи відкритий файл з кодом на мові «С». Закінчується аналіз у двох випадках:

– коли відсутні дані для аналізу;

– коли проаналізовано останній символ.

Після перевірки наявності вхідних даних і визначення того, чи є поточний символ останнім, розпочинається пошук відкриває круглої дужки, що є обов'язковою частиною кожної функції на мові «С». У випадку знаходження «((», аналізується структура перед дужкою, яка повинна мати вигляд: тип_функції ім'я_функції. Тип функції і ім'я функції можуть містити лише символи латинського алфавіту, цифри і символи підкреслення. Якщо функція оголошена правильно, вона додається до таблиці функцій, і визначаються її параметри (які завжди розміщені у круглих дужках).

Після цього триває пошук наступної функції, при цьому починається підрахунок фігурних дужок, які визначають вкладеність функцій. У разі виявлення вкладеної функції перевіряється її наявність серед оголошених функцій, так як кожна функція повинна бути оголошена перед використанням. Така перевірка відбувається для всіх записів таблиці, яка містить всі оголошені раніше функції. При успішному знаходженні, тільки що знайдена функція і функція, що відкрила фігурну дужку, визначаються як «дитина» і «батько» відповідно. Це відношення перевіряється на актуальність за допомогою таблиці відносин, в яку заносяться тільки унікальні відносини типу «дитина» – «батько». Якщо такого відношення в таблиці не виявлено, воно додається як новий запис у останній рядок.

Алгоритм завершується при досягненні останнього символу вихідного коду. З усього вищезазначеного можна вивести, що алгоритм узагальненому вигляді складається з одного великого циклу. Початок цього циклу визначається наявністю хоча б одного символу для аналізу, умовою завершення є досягнення останнього символу, а крок – один символ. Також у алгоритмі присутні два

невеликих цикли всередині великого, в яких перевіряється наявність тієї чи іншої функції (відношення функцій) в одній з двох (або в обох) таблиць.

3.3 Створення моделі мережі Петрі для алгоритму пошуку функцій

Розглянемо розроблену методику для аналізу алгоритмів, що ґрунтується на використанні мереж Петрі для моделювання алгоритму, оцінки впливу характеристик мережі на поведінку алгоритму та надання рекомендацій з його удосконалення. Для наочності візьмемо приклад алгоритму пошуку необхідної інформації у тексті (ім'я функції в коді) та обробки цієї інформації (запис імені, типу та параметрів у Базу Функцій). Усі операції, дії, обчислення та прийняття рішень здійснюються через переходи. Позиції вказують на поточний стан системи та значення змінних. Фішка у позиції вказує на готовність системи перейти до наступної інструкції. Кожна позиція має один вихідний перехід, за винятком позиції, яка передує прийняттю рішення; такі позиції мають два вихідних переходи, відповідно до істинного та помилкового значення предиката. Умовою визначення істинності предиката є значення аргументу-змінної. Фішка у мережі представляє потік оброблюваних даних; отже, таких фішок буде лише одна, оскільки доступ до змінної та її зміна надаються тільки одному потоку з метою безпеки та збереження цілісності інформації. У початковому стані фішка знаходиться у позиції P1. Зображений на рисунку 3.1 алгоритм, що реалізований мережею Петрі [9].

Вхідні дані для алгоритму пошуку функцій складаються з файлів вихідного коду на мові Сі. Результатом роботи алгоритму є знайдені функції, які зберігаються в кількох змінних, що містять інформацію про тип, ім'я та параметри функції. Після цього ці дані записуються в дві таблиці: таблицю функцій і таблицю відносин.

Алгоритм, зображений на рисунку 3.1, починає свою роботу з завантаження даних для обробки – файлу з вихідним кодом програми. Він завершується у двох випадках: коли файл для обробки не було завантажено або коли проаналізовано останній символ вихідних даних. У мережі Петрі всі події моделюються

переходами, тому операції над даними виконуються саме у переходах мережі. Події зазвичай мають по дві вихідні позиції. У цьому контексті моделюються умови if..else. Для активації конкретного переходу потрібні додаткові умови (позиції), стан яких визначає, який перехід буде активовано.

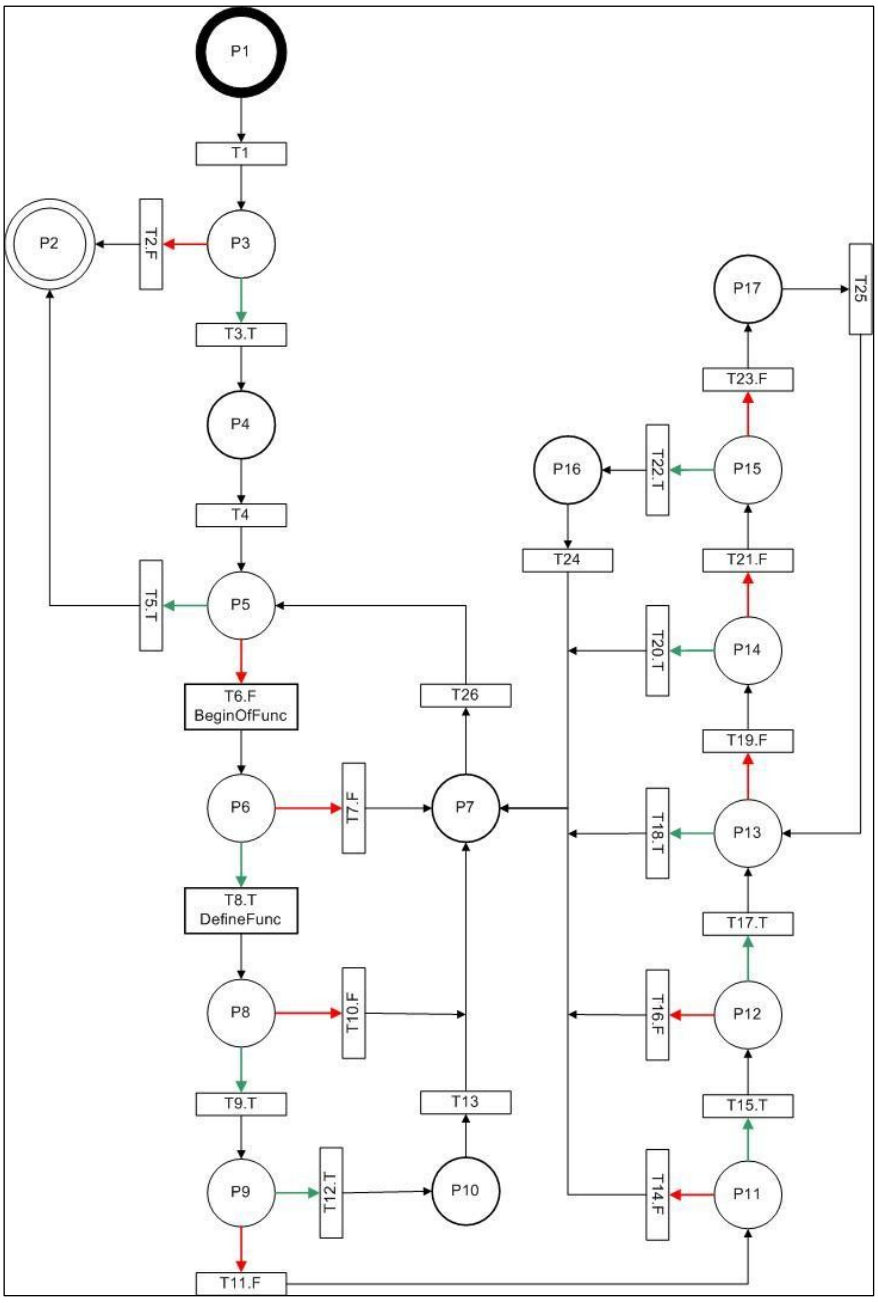


Рисунок 3.1 – Подання програмного алгоритму за допомогою мережі Петрі

Для полегшення зрозуміння мережі Петрі всі додаткові умови опущені, не відображаються на мережі. В іншому випадку це призвело б до повної втрати інформативності мережі, роблячи її практично несприйнятною. На прикладі рисунка 3.2 представлено випадок реалізації блоку «умова -> рішення» з та без

змінної «X». Як видно з діаграми, мережа може значно зростати навіть при відсутності відображення однієї змінної, не кажучи вже про дві або більше.

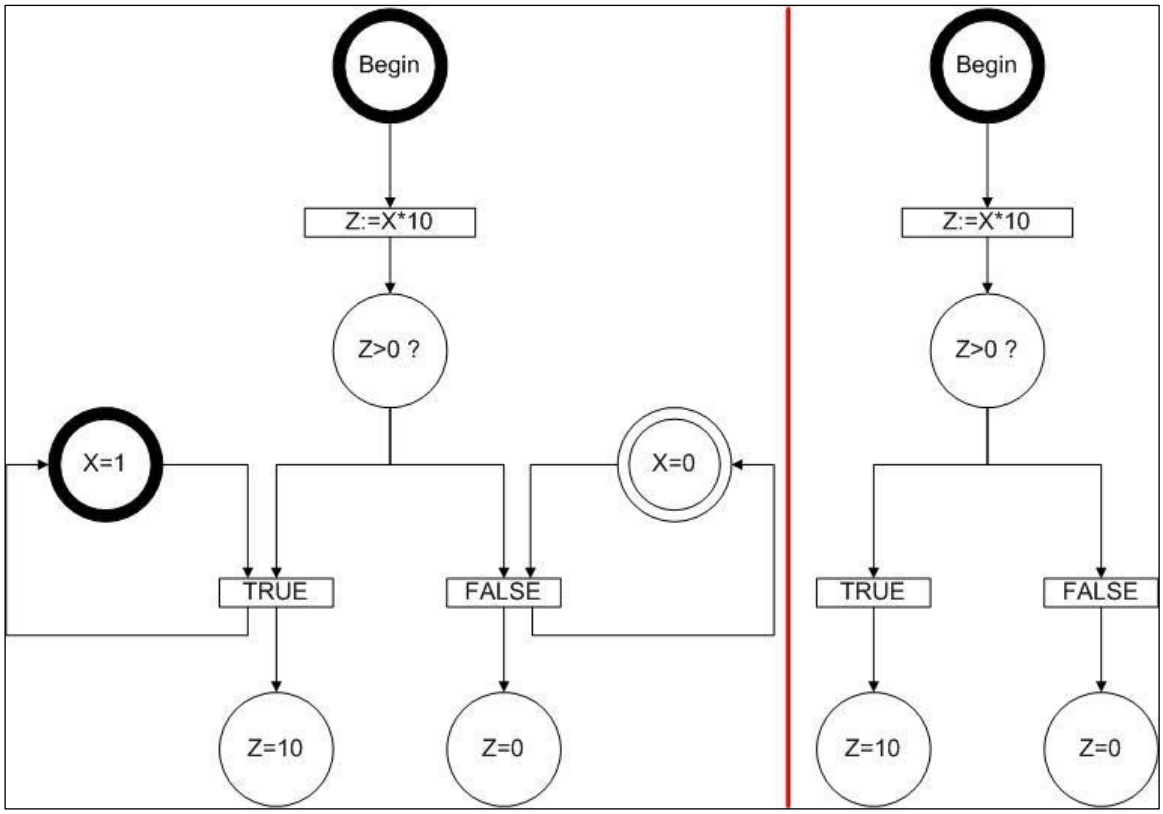


Рисунок 3.2 – Представлення змінної в мережі Петрі як умови для прийняття рішень

Давайте повернемося до алгоритму пошуку функцій і розглянемо значення кожної позиції та переходу. Позиція P1 є початковою, і фішка потрапляє в неї лише раз – під час запуску програми, тому ця позиція не має вхідних переходів. При виконанні переходу T1 відбувається завантаження файлу і очищення всіх таблиць, що містять результати пошуку. Позиція P3 перевіряє, чи завантажений вихідний код для аналізу. Якщо код завантажений, то відбувається перехід T3.T, де встановлюються змінні в початковий стан, коли ще не знайдено жодної функції. У випадку, якщо відкритий файл містить інформацію, несхожу на вихідний код програми, виконується перехід T2.F, і фішка потрапляє в позицію P2, що створює тупик у мережі – більше не можна виконати жоден перехід. Іншими словами, для продовження роботи з програмою необхідно відкрити інший файл, який містить код на мові C.

Перехід T4 забирає перший символ з текстового файлу для аналізу. У позиції P5 перевіряється, чи це останній символ. Якщо так, відбувається перехід до T5.T, а потім фішка знову переходить в позицію P2. T2.F і T5.T є єдиними переходами, що завершують обробку файлу в алгоритмі.

Перехід T6.F є складною подією. Складні події мають тривалість, відмінну від нуля, і можуть перетинатися в часі. Так як більшість подій у реальному світі займають час, вони є складними подіями, що не можуть бути адекватно представлені як переходи в мережі Петрі. Однак це не створює проблем при моделюванні систем. Складну подію можна представити як дві прості події: «початок складної події» і «закінчення складної події», з умовою «складна подія відбувається». Перехід T6.F представляє функцію BeginOfFunc, яка містить безліч умов і подій, але для спрощення мережі вона не розкладається на окремі складові, а представлена як складна подія. Ця функція виконує підрахунок відкритих дужок для визначення вкладеності функцій. Дії з підрахунку дужок виділені в окрему функцію, щоб в подальшому можна було розширити алгоритм для обробки не лише коду на Сі, а й на інших мовах, таких як Паскаль, де вкладеність визначається кількістю відкритих структур begin-end.

В позиції P6 перевіряється поточний символ на наявність «(», який є частиною будь-якої функції. При знаходженні цього символу можна припускати, що він належить до функції. Якщо поточний символ не є «(», тоді фішка переходить у позицію P7 через перехід T7.F, яка передує переходу T26. Перехід T26 переміщує покажчик на наступний символ оброблюваного файлу, після чого знову відбувається перевірка в позиції P5, чи не є цей символ останнім.

Дії з визначення імені функції виконуються через непрямий перехід-функцію T8.T, відому як «DefineFunc». Наступна позиція, P8, перевіряє, чи було визначено ім'я функції. Якщо ні, то фішка за допомогою переходу T10.F потрапляє у позицію P7, що означає готовність системи до аналізу нового символу. Якщо ж ім'я функції визначено (містить принаймні один допустимий символ), то відбувається перевірка правильності оголошення функції в позиції P9 (необхідно визначити тип і параметри, а також функція не може мати батьківської

функції). Якщо функція правильно оголошена, то вона додається в таблицю функцій через перехід T13. У випадку, якщо функція-батько не закрита, знайдена функція вважається вкладеною (P11, T15.T), і проводиться перевірка, чи були знайдені функція і функція-батько зареєстровані в базі даних відносин.

Алгоритм перевірки включає ітерації через таблицю вже знайдених відносин, починаючи з позиції T17.T, де покажчик встановлюється на перший рядок таблиці. Цей процес завершується одним з трьох можливих переходів:

- перехід до позиції T18.T відбувається, якщо поточний рядок таблиці є останнім;
- перехід до позиції T20.T відбувається, якщо відношення вже присутнє у базі даних;
- перехід до позиції T24 відбувається, якщо поточний рядок таблиці є передостаннім (останній завжди порожній), що означає, що нове відношення буде додане до таблиці.

Це описує алгоритм пошуку функцій, який був промодельований за допомогою апарату теорії мереж Петрі. Аналіз цієї моделі необхідний для отримання додаткової інформації. Мета аналізу полягає в тому, щоб дати відповідь на питання про конкретну мережу Петрі.

3.4 Дослідження моделі алгоритму пошуку функцій та особливості мереж Петрі

Аналіз моделі включає в себе ідентифікацію різних характеристик, зміна яких може призвести до зміни в роботі самої моделі. У цьому випадку очікується покращення продуктивності алгоритму пошуку після оптимізації характеристик мережі, описаних нижче.

3.4.1 Обмеження та забезпечення безпеки

Мережа Петрі вважається безпечною, якщо в кожній позиції максимально може бути лише одна фішка. Безпечна мережа Петрі передбачає безпечність усіх її позицій. Безпека є ключовою характеристикою для мережі, оскільки дозволяє

забезпечити безпечність позиції за допомогою одного біта пам'яті. Позиція вважається безпечною, якщо кількість фішок у ній для всіх переходів дорівнює 0 або 1. Якщо ж певний перехід має вихідну кратність 2 для деякої позиції, то при активації цього переходу до неї буде додано дві фішки, що робить її небезпечною. Якщо в певній позиції немає фішки, її значення вважається 0 (неправдою), а якщо є фішка, то значення становить 1 (правду).

Обмеженість є спеціальним випадком більш загальної властивості безпеки. Мережа Петрі вважається обмеженою, якщо існує таке ціле число k, що кількість фішок в будь-якій позиції не перевищує k. Позиція називається k-обмеженою, якщо кількість фішок у ній не перевищує ціле число k. У реалізації позиції використовується обмежений об'єкт, хоча сама структура мережі не відображає цього факту.

Обидва ці властивості можна перевірити за допомогою дерева досяжності. Мережа Петрі є обмеженою лише у випадку, коли у її дереві досяжності відсутній символ w. Якщо ж символ w присутній у дереві досяжності, це означає, що кількість фішок потенційно не обмежена. Така ситуація може призвести до послідовності запусків переходів, яку можна повторювати нескінченно, збільшуючи кількість фішок до довільно великого числа. Отже, наявність символу w свідчить про необмеженість мережі. Крім того, місцезнаходження символу w вказує на ті позиції, які також є необмеженими.

Це дозволяє зробити висновок, що необмежена позиція, що відображає ділянку програмного коду, є нескінченним циклом, що за певних значень змінних може призвести до «зациклення» програми та аварійного завершення. У випадку виявлення необмежених позицій на дереві досяжності слід переглянути програмний код, що вони представляють [10].

Мережа Петрі на рисунку 3.1 є безпечною, оскільки всі позиції мають вихідну кратність для будь-якого переходу рівну 0 або 1. У цьому випадку позиції не несуть жодного смислового навантаження, оскільки всі операції над даними відображаються переходами. Для спрощення мережі деякі позиції можна видалити, а два переходи об'єднати в один.

3.4.2 Збереження

Мережа Петрі вважається збереженою, якщо вона лише переміщує фішки, не втрачаючи їх або не створюючи нових. Це обмеження є досить сильним, оскільки воно вимагає, щоб кількість входів в перехід дорівнювала кількості виходів. Відхилення від цього правила призводило б до зміни кількості фішок у мережі, що може спричинити проблеми з доступом до даних і, відповідно, некоректною роботою програми.

Консервативність є важливою властивістю при моделюванні систем розподілу ресурсів в операційних системах. Мережа Петрі повинна зберігати ресурси, які вона моделює. Втрата фішки означає втрату цих даних, тоді як поява нової фішки може призвести до переповнення пам'яті новими даними. В обох випадках необхідно враховувати обмеження.

Важним висновком щодо цієї властивості є те, що у випадку, коли мережа Петрі, що моделює програмний продукт, є консервативною, можна розділити її на модулі, кожен з яких містить кілька позицій-переходів. Це можливо через те, що передбачено, скільки «фішок» прийде на вхід кожного модуля. Іншими словами, можливість моделювати програмний продукт з консервативністю дозволяє розбити його на декілька незалежних модулів, які можуть розроблятися різними групами програмістів. Наприклад, з мережі, зображеної на рисунку 3.1, можна виділити модуль, що перевіряє функціонал на актуальність, тобто шукає його вже доданим до бази; якщо не знаходить, то додає, інакше продовжує пошук нової функції.

Цей модуль починається з переходу T17 і в позицію P13, наступну за цим переходом, завжди надходить тільки одна фішка. Робота в цьому модулі завершується за одним з трьох переходів: T18, T20, T24; але лише один з них активується залежно від вхідних даних, тому в позицію основного модуля, P7, потрапляє лише одна фішка.

Таким чином, цей модуль можна об'єднати в один блок-перехід, який буде незалежним від решти мережі, оскільки на його вході та виході завжди буде постійна кількість фішок. Це дозволить ефективніше розподілити ресурси для

розробки окремих частин програми і, відповідно, зменшити час розробки продукту в цілому. Кількість входів і виходів кожного переходу становить |I(Tj)| = | O(Tj) | = 1. Це означає, що будь-який перехід лише переміщує фішку, не видаляючи її або не генеруючи нової. Отже, мережа Петрі на рисунку 3.1 є зберігаючою.

3.4.3 Активність

Під час розподілу ресурсів у операційних системах виникає проблема глухих шляхів для певних подій, як, наприклад, взаємні блокування процесів. У мережі Петрі глухі шляхи представлені переходами (або наборами переходів), які не можуть бути активовані. Перехід вважається активним, якщо він не заблокований (тобто не є глухим шляхом). Це не означає, що перехід обов'язково буде активований, але лише те, що він може бути активований. Перехід вважається активним, якщо існує маркування, при якому він може бути активований. Отже, якщо перехід активний, то завжди можливо перевести мережу Петрі з поточного маркування в маркування, при якому активування переходу буде можливим. Існує п'ять рівнів активності переходів. Наприклад, перехід з рівнем активності 0 вважається пасивним, тобто його ніколи не можна активувати. Перехід з рівнем активності 4 вважається активним, тобто його можна активувати при будь-якому маркуванні.

Ця властивість надзвичайно важлива для алгоритму моделювання. Із першого погляду на мережу Петрі можна визначити, які переходи активні, а які неактивні. Переходи з рівнем активності 4 вважаються вхідними для необмежених позицій і повинні бути уникнуті, оскільки можуть створити безкінечний цикл, якщо їх необмежено активувати, як вказано в розділі 3.4.1. Переходи з рівнем активності 0 вважаються вхідними для позицій, у які ніколи не потрапить фішка, що може призвести до невиконання функцій, пов'язаних із цими переходами. Такі переходи потрібно перевизначити або видалити.

На дереві досяжності перехід з рівнем активності 4 часто передує дублюючій вершині і вершині, яка містить маркер w. Пасивні переходи (з рівнем активності 0) на дереві досяжності відсутні і не відображаються.

У даній мережі відсутні пасивні переходи, кожен з них потенційно може бути активованим. Глухий шлях виникає тільки у випадку активації переходів T2 або T5, але тоді програма просто завершує аналіз даних, переносячи фішку до позиції P2 (Вихід).

3.4.5 Дерево досяжності

Дерево досяжності відображає всі можливі послідовності переходів у мережі Петрі. Кожна шлях у цьому дереві відповідає допустимій послідовності переходів, і вершина позначає маркування мережі. Гілки дерева вказують на зв'язки між різними маркуваннями мережі. Якщо деякі переходи можуть спрацьовувати нескінченну кількість разів, дерево досяжності стає нескінченним.

Побудова дерева досяжності допомагає встановити характеристики мережі і робить можливим зробити висновки щодо її оптимізації. Відсутність деяких переходів у дереві досяжності означає, що ці умови ніколи не будуть перевірені, і відповідно, подальші події не відбудуться.

Оскільки мережа Петрі є збереженою та безпечною, маркування мережі складається з шістнадцяти нулів та однієї одиниці у поточній позиції.

Наприклад, коли фішка перебуває в позиції P5, мережеве позначення виглядатиме так: (0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0). Таке позначення може бути складним для сприйняття, тому для більшої зрозумілості вершина позначається не позначенням, а позицією, де знаходиться фішка у даний момент. Дерево досяжності, зображене на рисунку 3.3, припускаючи, що кількість вхідних даних стандартна.

У цьому дереві вхідними даними є символи вихідного коду. Згідно з алгоритмом, вершина P5 зустрічатиметься стільки разів, скільки символів у вже згаданому тексті. Термінальні вершини виділені червоною рамкою, а дублюючі – синьою.

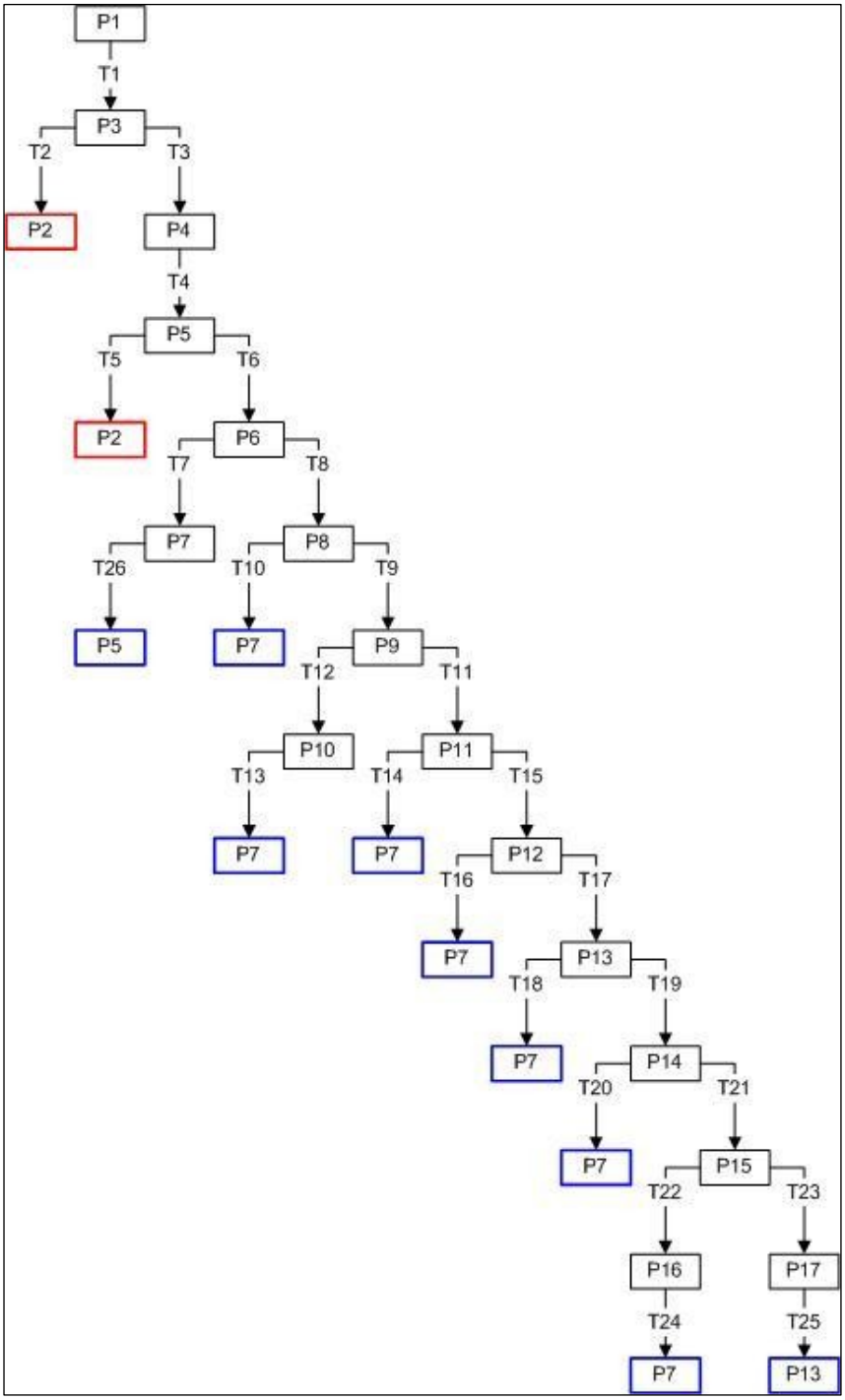


Рисунок 3.3 – Шляхи досягнення в мережі Петрі

Під час досліджень мережі Петрі як інструменту для моделювання та аналізу обчислювальних систем, було виявлено ряд властивостей моделі алгоритму пошуку, зміна яких значно вплинула на його поведінку та роботу. Наприклад, властивість активності дозволила виявити в мережі пасивні переходи, які моделювали неробочі частини алгоритму. Також були надані рекомендації

стосовно основних властивостей мереж Петрі, правильна оцінка яких допоможе уникнути «тупиків» і «вузьких місць» у алгоритмах.

Було розглянуто, як мережі Петрі можуть ефективно застосовуватися для формалізації алгоритмічних процесів і управління станами системи, забезпечуючи наочність і строгість в моделюванні програмних процедур. Такий підхід дозволив визначити ключові характеристики алгоритмів, їх взаємодії та потенційні точки оптимізації.

4 ОПИС ПРОГРАМНОЇ СИСТЕМИ

4.1 Мета програмної системи

Ця програмна система аналізує структуру програмних систем, написаних на мовах «C» або «C ++», виявляє функції у модулях та надає інформацію про їх залежності одна від одної у вигляді дерева з коренем у функції-батька. Її можуть використовувати всі, хто працює з комп'ютерним програмуванням.

Головною метою створення цієї системи є перевірка адекватності методики аналізу ефективності алгоритмів шляхом аналізу програмного коду. Це допомагає розробити основні принципи та кроки для оптимізації алгоритмів. Система надає інформацію про структуру аналізованого програмного комплексу у вигляді дерева з функціями як гілками, які були створені під час проектування програмного продукту.

Вона також вміє шукати специфічні функції, такі як рекурсивні або ті, що викликають одна одну, і відображає результати пошуку у вигляді таблиць, що полегшує знаходження потрібних функцій. Все це значно ускладнює алгоритм, що використовується для розробки методики аналізу алгоритмів за допомогою апарату теорії мереж Петрі, та надихає на більш детальне вивчення проблеми оптимізації алгоритмів.

4.2 Функціональні характеристики, які потрібно врахувати

Функції, які має виконувати дана програмна система, включають у себе:

- пошук функцій у вихідному коді;
- виключно пошук функцій, створених програмістом;
- відображення інформації про знайдені функції, включаючи їх тип, ім'я та параметри;
- встановлення відносин між функціями;
- подання цих відносин у вигляді таблиці;
- побудова дерева функцій для більш інформативного відображення структури програмного продукту;

– аналіз одного обраного користувачем файлу.
– аналіз проектів;
– визначення залежностей функцій у проекті;
– відображення фрагмента вихідного коду, що оточує знайдену функцію;
– показ прогресу пошуку;
– збереження результатів пошуку у файл, включаючи лише дерево функцій, якщо воно не порожнє.

4.3 Організація програмного продукту

У цій системі реалізовано алгоритм пошуку функцій з можливістю відображення результатів у вигляді дерева. Системну структуру можна розбити на модулі і блоки даних. Модулі – це частини програмного продукту, що відповідають за виконання певних функцій. На схемі 4.1 модулі представлені у вигляді овалів.

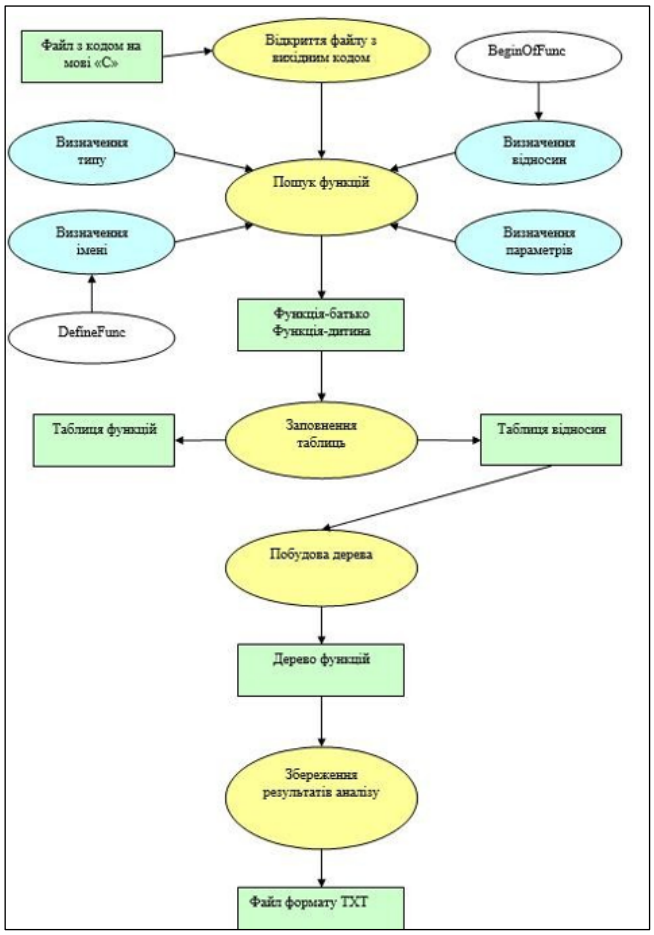


Рисунок 4.1 – Структура системи побудови дерева функцій

Модулі жовтого кольору (розташовані у центральному ряді) можуть розроблятися незалежно, оскільки заздалегідь відомо, який тип даних вони отримають на вхід. Наприклад, модуль побудови дерева потребує принаймні одного рядка в таблиці відносин, яка містить два об'єкти – ім'я батьківської функції та ім'я дочірньої функції. Детальніше модуль можна розглядати як набір підмодулів-функцій.

Інформація на рисунку 4.1 зображена у зелених прямокутниках, які служать початковими даними для одного модуля та вхідними для іншого, що є зв'язками всієї системи. Вхідними даними для програми є файли вихідного коду програм на мові «C» або «C++». Результатом є дві таблиці, дерево функцій та файл у текстовому форматі, який показує структуру дерева [11]. Програма має функцію перегляду вхідних даних у вікні головного вікна, як показано на рисунку 4.2.

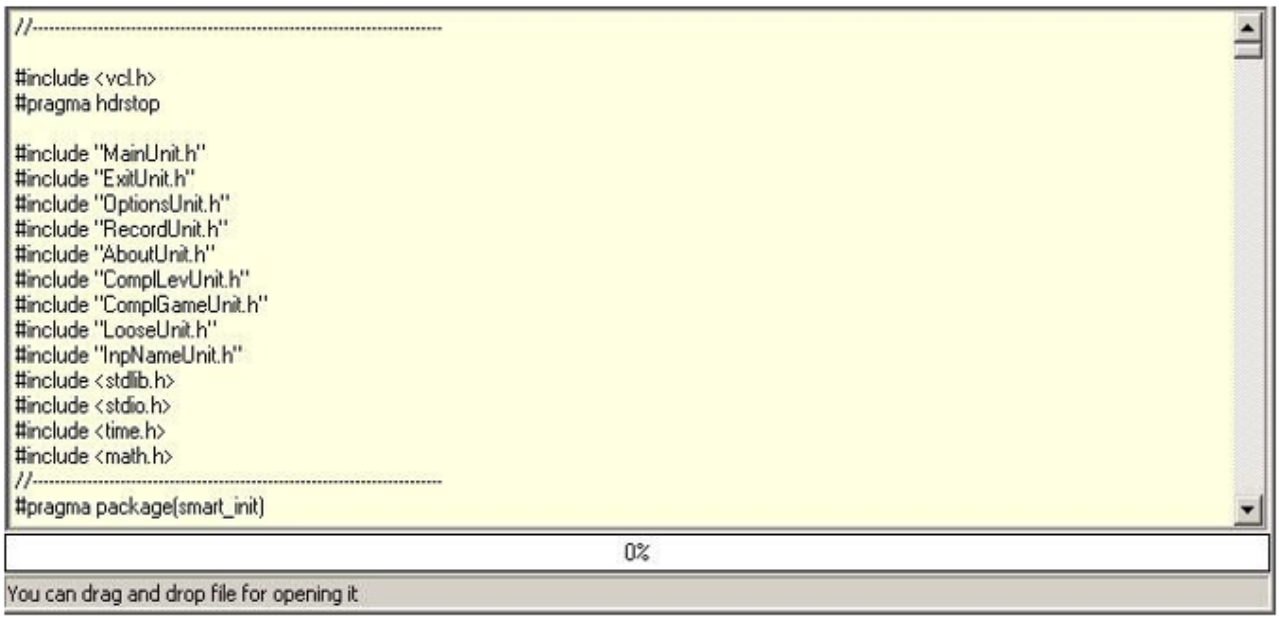
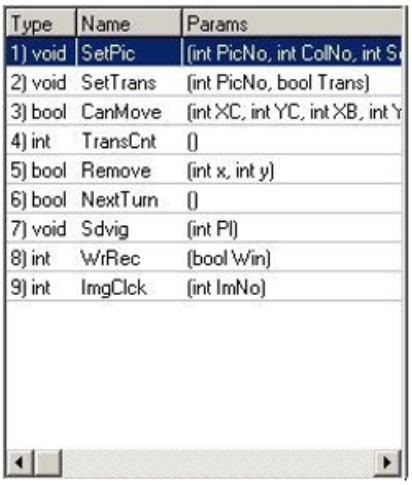


Рисунок 4.2 – Код програми, що був проаналізований

У першій таблиці є три колонки, що містять інформацію про знайдені функції: тип, який включає в себе тип функції та її порядковий номер для зручності; ім'я, що показує назву функції, надану програмістом; і параметри, що містять інформацію про прийняті параметри. На рисунку 4.3 представлена таблиця функцій з результатами пошуку.



Type	Name	Params
1) void	SetPic	(int PicNo, int ColNo, int S
2) void	SetTrans	(int PicNo, bool Trans)
3) bool	CanMove	(int XC, int YC, int XB, int Y
4) int	TransCnt	()
5) bool	Remove	(int x, int y)
6) bool	NextTurn	()
7) void	Sdvig	(int Pl)
8) int	WrRec	(bool Win)
9) int	ImgClck	(int ImNo)

Рисунок 4.3 – Таблиця функцій

Друга таблиця програми, яка називається «Таблиця відносин», містить інформацію про зв'язки між функціями батько-дитина. Вона складається з двох полів:

– дитина: це поле відображає ім'я функції, яка викликається всередині тіла батьківської функції;
– батько: це поле відображає ім'я функції, яка викликає дитячу функцію всередині свого тіла.

На рисунку 4.4 показана ця таблиця з вже заповненими даними під час аналізу вихідного коду.

Child	Parent
SetPic	Remove
SetTrans	Remove
CanMove	ImgClck
*CanMove	CanMove
TransCnt	Remove
Remove	NextTurn
NextTurn	ImgClck
SetPic	NextTurn
Sdvig	WrRec
WrRec	ImgClck
ImgClck	WinMain
SetPic	ImgClck
Remove	ImgClck
TransCnt	ImgClck
SetTrans	ImgClck

Рисунок 4.4 – Таблиця відносин

Після аналізування вихідного коду і виявлення хоча б однієї функції можна створити дерево функцій, де всі взаємозв'язки між ними будуть відображені у вигляді деревоподібної структури. Корінь цього дерева буде складатися з головної функції файлу або імені файлу, в залежності від того, чи аналізується весь проект чи лише окремий файл. На рисунку 4.5 вже зображено дерево функцій одного файлу.

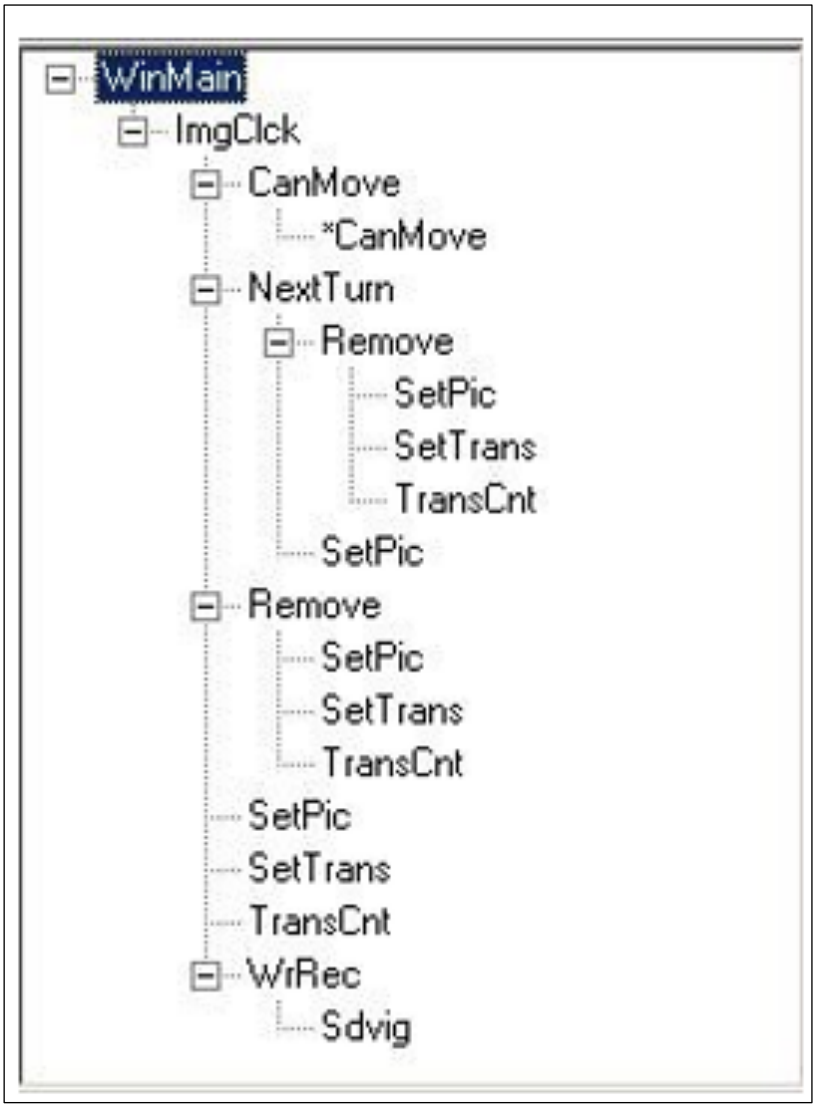


Рисунок 4.5 – Дерево функцій

На зображені 4.6 відображається інтерфейс програмної системи, який включає панель інструментів, де розташовані всі функції програми.

Рисунок 4.6 – Панель інструментів

Давайте розглянемо кожен елемент панелі інструментів зліва направо.

а) меню «File»: Цей пункт надає доступ до наступних опцій:

1) «Open»: Дозволяє відкрити будь-який файл для аналізу;
2) «Open project»: Відкриває вказаний користувачем файл проекту;
3) «Save»: Зберігає непорожнє дерево в поточний файл. Якщо файл не вказано, активується «Save as»;
4) «Save as»: Дозволяє зберегти дерево у новий файл або перезаписати існуючий;
5) «Exit»: Закриває додаток і звільняє пам'ять;

б) меню «Help»: Відкриває вікно «Про програму»;

в) значок «Open file»: Еквівалент пункту «Open» у меню «File»;

г) значок «Open project»: Те ж саме, що і «Open project» у меню «File»;

д) список «Project files»: Надає інформацію про файли проекту і дозволяє вибрати будь-який файл для аналізу. Усі файли впорядковані за алфавітом. Тестовий проект включає файли, зазначені на рисунку 4.7.

Рисунок 4.7 – Список файлів проекту

Значок «Аналіз проекту» дозволяє провести комплексний аналіз всього проекту. У цьому випадку пошук функцій відбувається в усіх модулях проекту, і структура дерева функцій виглядає так, як показано на рисунку 4.8. Варто зазначити, що коренем дерева є назви модулів проекту, а не назва головної функції, як у випадку аналізу одного файлу.

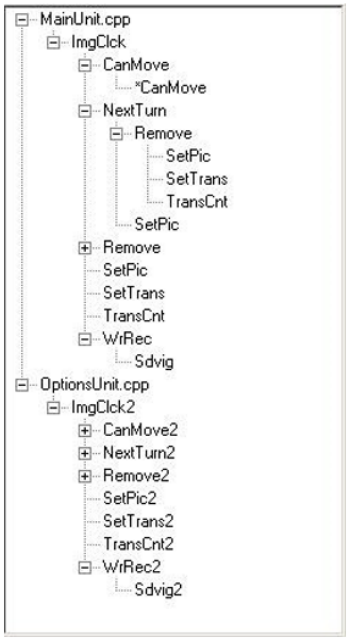


Рисунок 4.8 – Дерево функцій проекту

Натиснувши кнопку «Аналіз», автоматично проводиться аналіз та пошук функцій після завантаження коду. Якщо кнопка «Побудувати дерево» натиснута до аналізу вихідного коду, то автоматично створюється дерево функцій відповідно до результатів аналізу. При натисканні кнопки «Розгорнути дерево» воно автоматично розгортається повністю, в іншому випадку це можна зробити вручну.

Також додано наступні функціональні можливості:

- меню дерева, що дозволяє очистити його або зберегти у файл;
- виділення оголошення функції в файлі при натисканні на її ім'я в таблиці функцій;
- відображення прогресу аналізу файлу у відсотках від загального обсягу файлу;
- підрахунок кількості знайдених функцій;
- сортування вузлів дерева за алфавітом.

4.4 Тестування програмної системи побудови дерева функцій

Програму «Lines» тестували на основі вихідного коду. У даному контексті

суть програми «Lines» не має значення, а важливі лише ті характеристики, які впливають на результати тестування. Першою такою характеристикою є кількість модулів у проекті, яка становить 9 файлів:

- AboutUnit.cpp;
- ComplGameUnit.cpp;
- ComplLevUnit.cpp;
- ExitUnit.cpp;
- InpNameUnit.cpp;
- LooseUnit.cpp;
- MainUnit.cpp;
- OptionsUnit.cpp;
- RecordUnit.cpp.

Ще однією важливою характеристикою аналізованого вихідного коду є наявність та кількість функцій, які призначені для користувача. У цьому випадку такі функції визначаються та використовуються тільки в одному модулі проекту, а саме, в MainUnit.cpp. Їх кількість складає 9 штук.

Також слід звернути увагу на рівень вкладеності функцій у програмі. Найбільший рівень вкладеності в програмі «Lines» становить п'ятий: SetPic () -> Remove () -> NextTurn () -> ImgClck () -> WinMain ().

Після повного аналізу програмної системи було виявлено рекурсивну функцію CanMove (), позначену символом «*». Результати відповідають дійсності. Також проведено перевірку на нестандартні ситуації, наприклад, видалення оголошень функцій, що призвело до втрати їх в місцях використання [12].

Виявлено як теорія автоматів та мереж Петрі застосовується для формалізації та оптимізації процесу розробки, дозволяючи ефективно аналізувати алгоритми і виявляти ключові елементи програмного продукту.

ВИСНОВКИ

В рамках цієї кваліфікаційної роботи було розроблено модель програмної системи, засновану на теорії автоматів та мереж Петрі. Основна мета дослідження полягала у вивченні та оцінці ефективності використання цих теоретичних підходів для розробки програмного забезпечення.

По-перше, була створена мережа Петрі для моделювання алгоритмів і процесів всередині програмної системи. Цей процес включав етапи визначення позицій і переходів, а також їх зв'язування дугами для створення графічного представлення системи. Мережа Петрі дозволила візуалізувати та формалізувати паралельні процеси, що значно спростило аналіз і оптимізацію системи. За допомогою мережі Петрі були виявлені та усунені потенційні вузькі місця та конфлікти ресурсів, що підвищило загальну продуктивність і надійність системи.

По-друге, була застосована теорія автоматів для абстрагування логічних взаємодій всередині системи. У рамках цього етапу були розглянуті різні види автоматів, такі як детерміновані та недетерміновані скінченні автомати, автомати з магазинною пам'яттю і машини Тюрінга. Використання автоматів дозволило структурувати і формалізувати складні логічні процеси, що полегшило їх розуміння та аналіз. Автомати також використовувалися для верифікації та тестування різних аспектів системи, що сприяло забезпеченню її коректності та надійності.

У ході дослідження був проведений огляд теорії мереж Петрі, яка є потужним інструментом для моделювання процесів. В процесі роботи були проаналізовані основні аспекти теорії автоматів, які відіграють ключову роль у розробці програмних систем, забезпечуючи засоби для моделювання і аналізу алгоритмів. Це включало вивчення концепцій стану, переходів і алфавітів, що дозволило глибше зрозуміти структуру і поведінку систем.

У рамках обґрунтування вибору інструментів для моделювання були розглянуті сфери застосування мереж Петрі та їх розширення, такі як використання часу та кольорів. Були досліджені стохастичні мережі Петрі для обліку часових аспектів і кольорові мережі Петрі для моделювання різних типів

об'єктів і ресурсів. Ці розширення дозволили більш точно і гнучко моделювати складні системи.

Також у роботі була створена і протестована програмна система для побудови дерева функцій, що підтвердило ефективність використання теорії автоматів і мереж Петрі для моделювання і аналізу програмних систем. Були розглянуті алгоритми моделювання, включаючи алгоритми пошуку функцій, і розроблена модель мережі Петрі для цього програмного забезпечення. Проведене дослідження моделі алгоритму пошуку функцій дозволило виявити особливості мереж Петрі, такі як обмеження і забезпечення безпеки, активність і дерево досяжності.

Таким чином, виконане дослідження підтвердило, що використання теорії автоматів і мереж Петрі в процесі розробки програмного забезпечення є дієвим і обґрунтованим. Моделювання логічних взаємодій за допомогою автоматів дозволяє ефективно абстрагувати складність системи, тоді як мережі Петрі сприяють аналізу та оптимізації паралельних процесів. Результати дослідження можуть бути використані для покращення процесів розробки, зменшення ризиків і підвищення якості програмного продукту.

У ході виконання дипломної роботи були проаналізовані теоретичні основи теорії автоматів і мереж Петрі, їх роль у розробці програмних систем. Було встановлено, що застосування цих теорій дозволяє значно підвищити ефективність моделювання та аналізу складних програмних систем. У результаті дослідження була розроблена програмна система, заснована на мережі Петрі та теорії автоматів, яка продемонструвала високу ефективність у моделюванні та аналізі алгоритмів.

Мета роботи була досягнута, що підтверджує відповідність отриманих результатів поставленим завданням і цілям дослідження. Все відповідає вихідній постановці завдань і цілям, зазначеним на початку роботи: створення моделі програмної системи методами теорії автоматів і мереж Петрі, обґрунтування вибору інструментів моделювання, створення моделей алгоритмів і розробка програмної системи для побудови дерева функцій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Automata Theory / Geeks For Geeks. URL: https://www.geeksforgeeks.org/theory-of-computation-automata-tutorials/ (дата звернення: 12.05.2024).

2. Basics of Automata Theory / Anushsom Medium. URL: https://anushsom.medium.com/automata-theory-basics-for-dummies-by-a-dummy-a4f10d97314d (дата звернення: 17.05.2024).

3. Samuel Medina-Garcia, Joselito Medina-Marin, Oscar Montaño-Arango, Manuel Gonzalez-Hernandez and Eva Selene Hernandez-Gress Appl. Sci. 2023, 13(20), 11192; URL: https://doi.org/10.3390/app132011192

4. Basics of Automata Theory / CS Stanford. URL: https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/automata-theory/basics.html (дата звернення: 19.05.2024).

5. K.S. Cheung, K.O. Chow, A Petri Net Based Method for Refining Object Oriented System Specifications 187, 15 July 2007, Pages 161-172.

6. Bridle, J. S., & Mitchie, D., An adaptive array classifier for achieving multiple class discrimination in the context of robot planning. International Journal of Man-Machine Studies, - 1998, 10(3), 237-255. DOI: 10.1016/S0020-7373 (78) 80004-9.

7. Coloured Petri Net Model of Two-Phase Commit Protocol With Multiple Participants / M. Iwaniak, W. Khadzhynov // Реєстрація, зберігання і обробка даних. – 2013. – Т. 15, № 3. – С. 61-70.

8. Lau, H.C., Jiang, Z., Thien, L.C., Lee, C.K., Evaluating the efficiency of public transport systems: a data envelopment analysis approach. Transportmetrica A: Transport Science, - 2022, 18(10), 1125-1144. DOI: 10.1080/23249935.2022.202251.

9. Kiriy V.V, Sheiko, I., Petrova, R., Optimization of management information support as a basis for organizational transformations at an enterprise, Periodicals of Engineering and Natural Sciences. 2019. V.7. N. 2. P. 679-689

10. Кравець Н. С., Інтерактивна імітація та аналіз розкрашеної сітки Петрі з використанням алгебри предикатних операцій, Праці п'ятої всеукраїнської міжнародної конференції 27 листопада -1 грудня 2000р.,Україна,Київ. С.313-316.

11. Shubin I.Yu, A. Kozyriev, V. Liashik, G. Chetverykov, Methods of Adaptive Knowledge Testing Based on the Theory of Logical Networks, Proceedings of the 5th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2021), Kharkiv, Ukraine, April 23-24, 2021. – Volume I, P. 1184-1193. (CEUR, SCOPUS)

12. Лановой О. Ф., Візуалізація в методах тестування програмного забезпечення, Харків, 6-а Міжнародна науково-технічна конференція «ІСТ-2017», ХНУРЕ, 11-16 вересня 2017 р., С.110-111.