КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота

на здобуття ступеня бакалавра
за спеціальністю
121 Інженерія програмного забезпечення
на тему:

Розв'язання задачі розбиття множини цілочисельних векторів автоматними засобами

Виконав:
студент 4-го курсу
Павло КРАВЧУК _____
(підпис)

Науковий керівник:
професор, доктор фізико-математичних наук
Сергій КРИВИЙ _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
«25» травня 2022р.
Протокол № 10

Завідувач кафедри
Олександр ПРОВОТАР _____
(підпис)

Київ - 2022

РЕФЕРАТ

Обсяг роботи 41 сторінка, 10 джерел посилань, 2 додатки.

АВТОМАТИ, ВІЗУАЛІЗАЦІЯ АВТОМАТІВ, ВЕКТОРИ, ГІПЕРПЛОЩИНА, РОЗБИТТЯ.

Об’єктом роботи є теорія автоматів, а саме можливість її фактичного використання для розв’язання задач розбиття множини векторів за поставленими умовами. Предметом роботи є програмна система, що має можливість будувати та використовувати автомати, реалізовані з використанням певної мови програмування, будуючи їх за певним алгоритмом, а також візуалізувати процес розв’язання задачі, з використанням побудованого автомата.

Метою роботи є створення програмної системи, що будує автомати, розв’язує відповідні задачі та візуалізує процес розв’язку в реальному часі.

Інструменти розроблення: редактор текстових файлів та програмного коду Visual Studio Code, інтерпретатор мови програмування Python, оболонка бібліотеки користувацького інтерфейсу Qt для Python – PyQt, пакет інструментів для візуалізації графів Graphviz.

Результати роботи: проведене дослідження алгоритмів створення автоматів для розв’язку задач розбиття множини векторів, реалізовано ці алгоритми мовою програмування Python, створено програмну систему що розв’язує та візуалізує процес розв’язку задач розбиття векторів автоматними засобами.

Створена програмна система успішно розв’язує задачі побудови автоматів для розв’язку, та, власне, розв’язку задач розбиття множини цілочисельних векторів відносно окремого порогового вектора та відносно заданої гіперплощини. Подальший розвиток дослідження та розробки могли б полягати у розгляді інших методів розбиття множини векторів або розширені області дослідження до інших типів задач, з використанням автоматних засобів.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об’єкта розробки. Теорія скінчених автоматів почалась ще в 1943 році[1], які були формалізовані у 1955-1956 роках інформатиками Джоржем Мілі та Едвардом Форрестом Муром, які описали два типи скінчених автоматів, що були названі за їх іменами[2][3]. З того часу було розроблено багато теорій, алгоритмів та варіацій автоматів для розв’язку різних задач з їх використанням.

Одна з таких задач є задача розбиття множини векторів. З дотичних до цієї задачі тем вже існує низка досліджень[4], які стали підґрунтям для створення алгоритмів реалізації автоматів, використаних в цій роботі

Актуальність роботи та підстави для її виконання. Задача розбиття множини векторів не є розповсюдженою, а тому стала об’єктом дослідження цієї роботи та основою реалізованої програмної системи.

Мета й завдання роботи. Метою роботи є створення програмної системи, що має можливість будувати автомати для розв’язку задач розбиття множини векторів та використовувати їх, додатково виводячи процес розв’язку у користувацькому інтерфейсі. Для розроблення такої системи поставлені такі задачі:

— Дослідити алгоритми побудови автоматів для розв’язку задач розбиття векторів та інших алгоритмів, що будують автомати для розв’язку задач, дотичних до поставлених в роботі.

— Визначити підходи до реалізації алгоритмів з використанням певної мови програмування

— Розробити програмну систему для розв’язку задач розбиття цілочисельних векторів автоматними засобами та візуалізації процесу розв’язку цієї задачі.

Об'єкт, методи й засоби розроблення. Об’єктом розроблення програмної системи є реалізація певних алгоритмів для розв’язку задач розбиття векторів, використання їх для розв’язку відповідних задач та візуалізація процесу розв’язку в реальному часі.

Основний метод розроблення програмної системи – метод частинних цілей: система розбита на певну кількість модулів, кожна з яких має окремі задачі, які в поєднанні розв’язують поставлені задачі.

Для розроблення програмної системи були використані редактор текстових файлів та програмного коду Visual Studio Code, інтерпретатор мови програмування Python, оболонка бібліотеки користувацького інтерфейсу Qt для Python – PyQt, пакет інструментів для візуалізації графів Graphviz, пакет-оболонка взаємодії програмного коду Python та Graphviz – PyGraphviz, бібліотека Python для дослідження графів та мереж – NetworkX.

РОЗДІЛ 1 ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

1.1 Бібліотека для створення графічних застосунків Qt

Одна з цілей, поставлених для виконання в цій роботі – візуалізація виконання розв’язку задач розбиття векторів з використанням автоматів. Для самого процесу створення зображення автомата було обрано Graphviz та деякі допоміжні засоби, але без використання засобів створення та відображення графічного інтерфейсу було б необхідно вводити параметри створення автомата та елементи множини для розбиття використовуючи командний рядок, та було б важко виконувати динамічне відображення стану автомату, завдяки якому розв’язується задача. З цих причин використовується PyQt – оболонка кросплатформної бібліотеки Qt, що використовується для створення програм з графічним інтерфейсом.

Для створення графічного інтерфейсу Qt надає доступ до великої кількості віджетів[5], компонентів графічного інтерфейсу, частина з яких стали стандартом для такого виду бібліотек, такі як кнопки, поля вводу тексту, мітка, та інші. Інша частина компонентів зустрічаються рідше. Один з таких компонентів є QSvgWidget, який є основним для створення системи динамічного зображення стану скінченого автомата, який розв’язує певну задачу.

QSvgWidget, як походить з назви, відображає векторні зображення, описані у форматі SVG (Scalable Vector Graphics) на вікно графічного інтерфейсу програми. Формат SVG є підмножиною мови розмітки XML[6], яка має широке застосування і редагується більшістю мов програмування, як у стандартних бібліотеках таких мов, так і завдяки бібліотекам, створеними окремо. Такий підхід відображення стану автомату дозволяє швидко змінювати незначну частину зображення, у випадку створеної програмної системи, виділення поточного стану або переходу автомата на фоні інших, для візуалізації процесу розв’язку задачі.

1.2 Візуалізація автомата з використанням Graphviz

Створення зображень автомата є нетривіальною задачею. Для створення системи візуалізації потрібно мати можливість створення зображень, і для розв'язку цієї задачі був використаний пакет інструментів візуалізації графів Graphviz. Сам по собі цей пакет являє собою як готові виконувані файли, так і набір бібліотек та заголовків до них для використання у програмах, написаних мовою C. Для взаємодії програмної системи та пакету інструментів Graphviz використовується пакет-оболонка PyGraphviz[7], яка надає інструменти роботи з Graphviz у вигляді об'єктів та функцій.

Graphviz при безпосередньому використанні оброблює дані про граф, описанні мовою опису графів DOT. Приклад опису автомата мовою DOT наведено у додатку А.

Серед виконуваних файлів пакета наявні декілька файлів, що оброблюють графи у форматі DOT. Кожен з цих виконуваних файлів реалізований за певною теорією розміщення графів на площині, тобто створюють візуалізацію одного і того ж графу у різному вигляді. Серед наявних алгоритмів[8]: dot – що реалізує ієрархічну систему розміщення; neato – що реалізує пружинну модель розміщення, яка керується мінімізацією функції глобальної енергії зображення графа; fdp – що реалізує пружинну модель розміщення, яка керується мінімізацією сил, на відміну від neato тощо.

1.3 PyGraphviz та NetworkX

Бібліотека NetworkX[9] - одна з декількох бібліотек, що дозволяють створювати представлення графів засобами мови програмування Python та переводити це представлення до мови опису графів DOT, яке можна використовувати для створення фінального зображення графу через Graphviz, для чого як залежність використовується PyGraphviz.

Перевага NetworkX над подібними засобами полягає у можливості легкого встановлення параметрів відображення вузлів та зв’язків графу, або в такому випадку станів та переходів автомата, серед яких ідентифікатори об’єктів для подальшої обробки, відстань між переходами тощо.

РОЗДІЛ 2 АВТОМАТИ. ОПИС АЛГОРИТМІВ РОЗВ'ЯЗКУ ЗАДАЧ РОЗБИТТЯ

2.1 Детермінований скінченний автомат

В даній роботі розглядаються задачі розбиття множини векторів відносно порогового вектора та розбиття відносно гіперплощини. Тип автомата що розв'язує ці типи задач є детермінований скінченний автомат (ДСА), оскільки кінцевий розв'язок задачі не залежить прямо від входу що подається до автомата, а залежить лише від кінцевого стану (допустимого або не допустимого), на якому зупиняється автомат після завершення обробки входу. Наведемо формальне означення ДСА.

ДСА може бути визначений як кортеж з 5 елементів:

$$(Q, \Sigma, \delta, q_0, F) ,$$

де Q – множина станів,

Σ – скінченна множина вхідних символів,

δ – функція переходу ($\delta : Q \times \Sigma \to Q$)

q_0 – початковий стан, елемент множини Q

F – набір допустимих станів

Умовою прийняття вхідної строки є наявність переходів, починаючи з початкового стану, згідно з символами вхідної строки, що приводять до допустимого стану. Формально цю умову можна описати в 3 кроки:

— $r_0 = q_0$

— $r_{i+1} = \delta(r_i, a_{i+1})$, для кожного $i = \overline{0, n}$

— $r_n \in F$

де $a_1, a_2, a_3 \dots a_n$ – символи із множини Σ, що визначають проміжні стани

r_0 – початковий стан,

$r_1, r_2, r_3 \dots r_{n-1}$ – проміжні стани із множини Q

r_n – кінцевий стан для вхідних даних, для прийняття строки має належати множині F

2.2 Алгоритми задач розподілу множини векторів

2.2.1 Розбиття векторів відносно порогового вектора

Дамо формальне визначення задачі розбиття векторів відносно порогового вектора[10].

Для входу подається скінченна множина векторів $V = \{v_1, v_2, ..., v_m\}$ розмірності $n \in N$ та пороговий вектор $a = (c_1, ..., c_n)$ де $v_i, a \in Z^n$, Z – множина всіх чисел. Для порівняння векторів задається відношення порядку. В цій роботі розглядається покоординатний порядок, який є частковим.

Для двох векторів $x = (x_1, ..., x_n), y = (y_1, ..., y_n) \in Z^n$ покоординатне відношення порядку визначається за правилом:

$$x \leq y \leftrightarrow \forall i = \overline{1, n}, x_i \leq y_i$$

Тобто вектор x вважатиметься меншим або рівним вектору у якщо кожна координата вектора x менша або рівна відповідній координаті вектора у. При чому, для строгої нерівності векторів $x < y$ для хоча б одного з координат векторів повинна виконуватись нерівність $x_i < y_i$, де $i = \overline{1, n}$.

Задача розбиття полягає, у випадку покоординатного відношення порядку, у розбитті множини векторів на чотири підмножини: $V_1 = \{v \in V \mid v < a\}$, $V_2 = \{v \in V \mid v > a\}, V_3 = \{v \in V \mid v = a\}, V_4 = \{v \in V \mid v \propto a\}$, де символ $\propto$ означає непорівнянність векторів між собою.

2.2.1.1 Випадок цілих невід'ємних координат

Для розв'язку задачі розбиття з використанням автоматів необхідно побудувати автомат, що за допустимими станами зможе визначати належності вхідного вектора $v \in V$ до однієї з категорій-підмножин V_1, V_2, V_3, V_4. Одним з переваг розв'язку задачі з використанням автоматів полягає у можливості знаходження відношення між координатами $x_i < y_i$, $x_i = y_i$, $x_i > y_i$ за одну «операцію» порівняння. Наведемо приклад задачі що розв'язує такий автомат.

Візьмемо для порівняння вектори $v = (1, 4)$ та $a = (1, 2)$.

Для обробки автоматом координати векторів представляються у вигляді двійкових слів, на вхід автомата подаватимуться пари символів двійкових слів, що кодують координати векторів v та a одного розряду, починаючи з найстаршого. Отже, координати даних векторів необхідно перевести до двійкової системи числення:

$$v^T = \begin{pmatrix}1\\4\end{pmatrix} = \begin{pmatrix}0 & 0 & 1\\1 & 0 & 0\end{pmatrix} = v_3 v_2 v_1;\ a^T = \begin{pmatrix}1\\2\end{pmatrix} = \begin{pmatrix}0 & 0 & 1\\0 & 1 & 0\end{pmatrix} = a_3 a_2 a_1$$

$$v_3 = \begin{pmatrix}0\\1\end{pmatrix}, v_2 = \begin{pmatrix}0\\0\end{pmatrix}, v_1 = \begin{pmatrix}1\\0\end{pmatrix};\quad a_3 = \begin{pmatrix}0\\0\end{pmatrix}, a_2 = \begin{pmatrix}0\\1\end{pmatrix}, a_1 = \begin{pmatrix}1\\0\end{pmatrix}$$

Виходячи з вище сказаного визначимо вхідний алфавіт автомата, що розв’язує задачу розбиття:

$$x_1 = \begin{pmatrix}0\\0\end{pmatrix}, x_2 = \begin{pmatrix}0\\1\end{pmatrix}, x_3 = \begin{pmatrix}1\\0\end{pmatrix}, x_4 = \begin{pmatrix}1\\1\end{pmatrix}$$

В кожному із символів-векторів перша координата відповідає цифрі двійкового представлення вектора, що порівнюється, а друга координата цифрі із порогового вектора. Для представлення цих символів у програмній системі використовуватимуться рядковий тип даних, тому представлення цих символів буде наступним:

$$x_1 = \text{"0:0"}, x_2 = \text{"0:1"}, x_3 = \text{"1:0"}, x_4 = \text{"1:1"}$$

Тобто так само розміщені перша та друга координати символів-векторів, проте розділені двокрапкою.

Визначимо слова, що подаватимуться на вхід автомата, при розв’язку даного прикладу:

$$p_1 = \begin{pmatrix}0 & 0 & 1\\0 & 0 & 1\end{pmatrix} = x_1 x_4 x_4\ ;\ p_2 = \begin{pmatrix}1 & 0 & 0\\0 & 1 & 0\end{pmatrix} = x_3 x_2 x_1$$

Тепер розглянемо представлення автомата на рис. 2.1

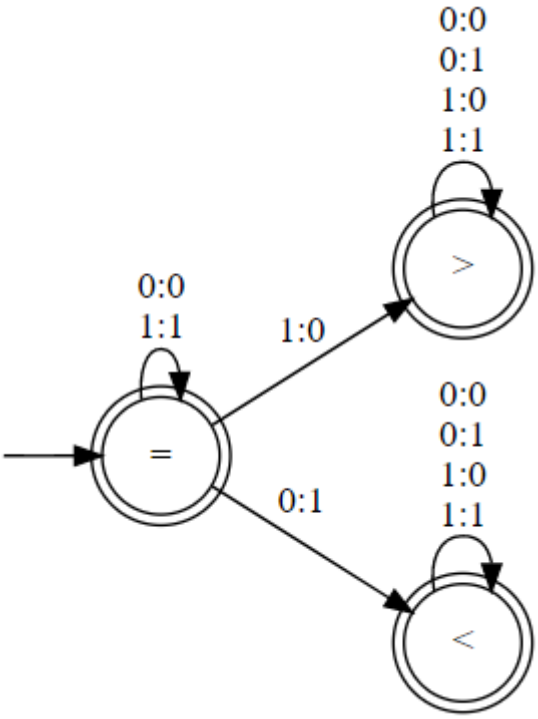


Рис 2.1

В цьому автоматі можемо бачити що всі стани заключні, початковий стан автомата являє собою стан рівності між координатами.

Аналіз автомата показує що він правильно визначає відношення порядку між координатами векторів: автомат починає роботу зі стану рівності, і залишається в ньому до тих пір, поки старші розряди двійкового представлення векторів рівні між собою. Як тільки знаходиться пара відмінних цифр цього представлення одразу відбувається перехід до стану нерівності. Оскільки при цьому переході розглядаються найстарші біти координат – вони й будуть найбільш значущі у цьому порівнянні, і більше зміни станів відбуватись не буде.

У результаті роботи автомата над наведеним раніше прикладом отримаємо результат порівняння координат векторів: для входу p_1 що порівнював координати 1 та 1 заключним станом буде рівність, для входу p_2 що порівнював координати 4 та 2 заключним станом буде знак більше. Отже, заключні стани цього розв'язку: $(=, >)$.

Згадуючи визначення покоординатного відношення порядку можемо зробити висновок що вектор v більший ніж пороговий вектор a, адже за означенням вектор

x вважається більший ніж вектор y у випадку коли кожна відповідна координата x нестрого більша за координату y, при чому принаймні одна координата строго більша за відповідну іншу, а отже вектор v належить підмножині V_2.

Розглянемо всі випадки результату роботи автомата та алгоритм визначення за цими результатами відношення порядку між векторами:

Нехай $K = \{k_1, \dots, k_n\}$ – заключні стани автомата після покоординатного порівняння векторів x та y.

— $\forall i, k_i \in \{=\} \leftrightarrow x = y$

— $\exists i \nexists j, k_i \in \{>\}\ k_j \in \{<\} \leftrightarrow x > y$

— $\exists i \nexists j, k_i \in \{<\}\ k_j \in \{>\} \leftrightarrow x < y$

— $\exists i \exists j, k_i \in \{<\}\ k_j \in \{>\} \leftrightarrow x \propto y$

Варто зазначити що для розбиття використовується один автомат, який послідовно порівнює координати попарно між собою. Для наочності програмної системи був використаний саме такий метод. Проте є можливість пришвидшити процес розбиття якщо використовувати мережу автоматів, в якій кожен автомат буде порівнювати свою пару координат. Для розуміння такого підходу введемо поняття мережі автоматів:

Мережею автоматів називається кортеж $A = (A_1, A_2, \dots, A_n)$ автоматів, де $A_i = (\mathrm{A}_i, X_i, f_i, a_{0i}, F_i), i = \overline{0, n}$. Символ $x \in X = X_1 \cup \dots \cup X_n$ називається дією в мережі. Конфігурацією в мережі A називається кортеж станів $(a_1, \dots, a_n)$ де $a_i \in \mathrm{A}_i,\ i = \overline{0, n}$. Конфігурація називається початковою, якщо $a_i = a_{0i}$, або заключною, якщо $a_i \in F_i, i = \overline{0, n}$.

На відміну від випадку для задачі розбиття автомати $A_1, A_2, \dots, A_n$ не обов’язково однакові, також алфавіти цих автоматів не обов’язково однакові. Тому результат дії над автоматами визначається наступним чином:

Нехай $A = (A_1, A_2, \dots, A_n)$ мережа автоматів, де $A_i = (\mathrm{A}_i, X_i, f_i, a_{0i}, F_i)$. Для деякої дії x автомат A_i бере участь в цій дії якщо $x \in X_i$. Дія x можлива на конфігурації $(a_1, \dots, a_n)$ якщо $f_i(a_i, x) = \emptyset$ виконується для всіх автоматів мережі,

що беруть участь в цій дії. Якщо дія x можлива, то вона виконується і генерує перехід з поточної конфігурації мережі до наступної конфігурації $(a'_1, \dots, a'_n)$

де $a'_i = \begin{cases} f_i(a_i, x), & \text{якщо } A_i \text{ бере участь в } x \\ \{a_i\}, & \text{інакше} \end{cases}$

Виконанням мережі A на вхідній послідовності дій $x_0 x_1 \dots x_{n-1} \in F(X)$ називається послідовність

$$c_0 \xrightarrow{x_0} c_1 \xrightarrow{x_1} c_2 \dots \xrightarrow{x_{n-1}} c_n$$

така, що c_i є конфігурацією для кожного $0 \le i \le n$, c_0 – початкова конфігурація, $f(a_i, x_i) = a_{i+1}$ для кожного $0 \le i \le n-1$.

Виконання називається допустимим, якщо c_n заключна конфігурація. Мережа A допускає слово $p = x_0 x_1 \dots x_{n-1} \in F(X)$, якщо вона має допустиме виконання на вхідній послідовності p.

2.2.1.2 Випадок цілих координат

Наступною частиною розв'язку поставленого завдання стане розширення поля можливих значень координат до множини всіх цілих чисел. Для цього необхідно змінити представлення координат векторів, щоб враховувати знак координати та модифікувати автомат що розв'язує задачу розбиття з урахуванням знаку координат.

Для цього введемо означення 2-доповнення двійкового числа x:

— Якщо $x \ge 0$, то $x_n = 0$ і $(x)_2 = 0 x_{n-1} \dots x_1 x_0$

— Якщо $x < 0$, то $x_n = 1$ і $(x)_2 = 1\overline{x_{n-1} \dots x_j} x_{j-1} \dots x_1 x_0$

де x_n – знаковий біт,

j – наймолодший розряд двійкового представлення де $x_j = 1$

$\overline{x_i} = 1 - x_i$, де $j < i \le n-1$

В результаті отримуємо представлення числа, незмінне від попереднього представлення у випадку невід'ємного значення, за виключенням додаткового нульового знакового біту, та у випадку від'ємного значення представлення що

зменшується у двійковому вигляді у випадку збільшення абсолютного значення що представляється, і додаткового знакового біту встановленого в значення один.

Таке представлення дає можливість правильно порівнювати координати між собою у випадку коли знакові біти рівні, порівняння координат у випадку різних знакових бітів тривіальне, якщо дещо змінити заданий раніше автомат (рис 2.2).

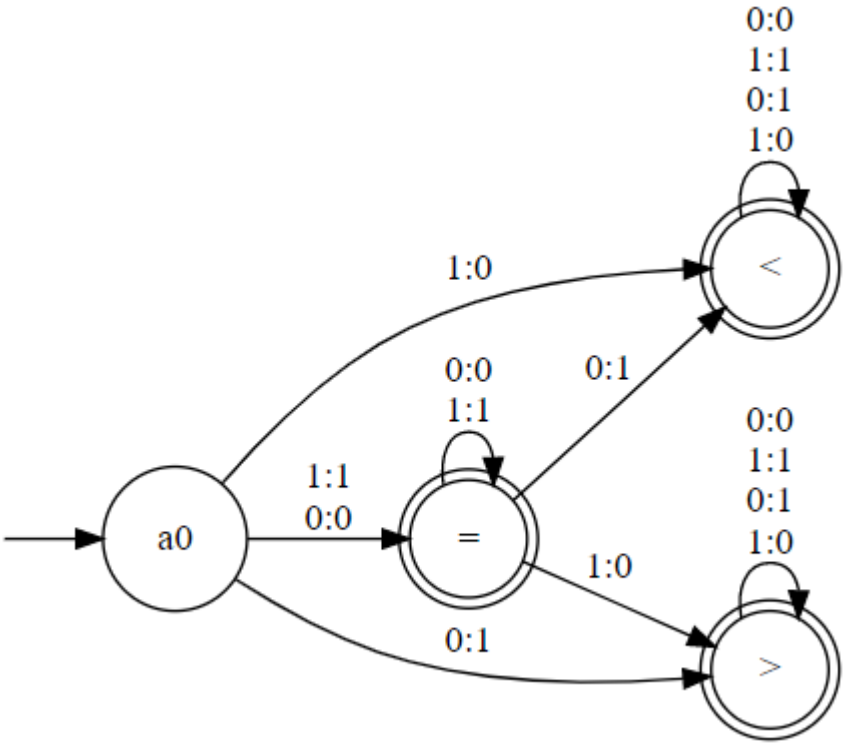


Рис 2.2

З рисунка автомата можемо бачити новий стан. Цей стан відповідає за вирішення результату порівняння в залежності від знакових бітів. У випадку різних бітів одразу відбувається перехід до відповідного стану нерівності. Інакше автомат працює за алгоритмом без розширення, ви виходячи зі сказаного вище – цей алгоритм працює правильно як для двох від’ємних, так і двох додатних координат.

2.2.2 Розбиття векторів відносно гіперплощини

Наступна задача яку розв’язуватиме програмна система буде задача розбиття векторів відносно гіперплощини, що задана рівнянням $a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$. Для розв’язку такої задачі потрібно розглянути методи розв’язку цієї

формули, що є формулою арифметики Пресбургера (АП), з використанням автоматів.

Перед початком розгляду цих розв'язків введемо поняття арифметики Пресбургера:

В АП використовується алфавіт змінних $X = \{x, y, \dots, z, x_1, y_2, \dots\}$, константи 0 та 1 і бінарна операція +. Термом АП називаються:

— Довільна змінна із X або константи 0, 1;

— Якщо t_1 і t_2 – терми АП, то $t_1 + t_2$ – терм АП

Також в АП допустимі скорочення, прийняті у звичайній арифметиці, наприклад формула $x + y + y + z + z + z + 1 + 1$ може бути записана у вигляді $x + 2y + 3z + 2$.

Атомарною формулою АП називається рівність або нерівність між двома термами АП, тобто виразі $t_1 = t_2$ та $t_1 \geq t_2$, де t_1, t_2 – терми АП.

2.2.2.1 Визначення нерівності у випадку невід'ємних чисел

Розглянемо алгоритм побудови автомата, що приймає розв'язки лінійного обмеження, описаний такою атомарною формулою АП:

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b$$

Знову будемо розглядати розв'язування задач розподілу векторів, при попередньому перетворені їх до двійкової системи числення. Проте цього разу цифри у двійковій системі числення записуватимуться за збільшенням розрядності. Наприклад вектор $a = (10, 14)$ буде представлений у двійковому вигляді наступним чином:

$$a^T = \begin{pmatrix}10\\14\end{pmatrix} = \begin{pmatrix}0 & 1 & 0 & 1\\0 & 1 & 1 & 1\end{pmatrix} = a_1a_2a_3a_4$$

$$a_1 = \begin{pmatrix}0\\0\end{pmatrix}, a_2 = \begin{pmatrix}1\\1\end{pmatrix}, a_3 = \begin{pmatrix}0\\1\end{pmatrix}, a_4 = \begin{pmatrix}1\\1\end{pmatrix}$$

Також надалі записуватимемо наведену раніше атомарну формулу АП скорочено у вигляді $(a, x) \leq b$, де (a, x) це скалярний добуток векторів $a = (a_1, a_2, \dots, a_n)$ та $x = (x_1, x_2, \dots, x_n)$.

Тепер розглянемо приклад покрокового розв’язку формули у двійковій системі числення, в чому полягатиме алгоритм побудови автомата для розв’язку певної формули АП, що описана раніше:

Нехай формула АП що розглядається має вигляд:

$$x_1 + 2x_2 - 3x_3 \leq 2$$

Розв’язки що будуть перевірятися на виконуваність обмеження:

$$v_1 = (1, 2, 1), v_2 = (1, 2, 2)$$

Переведемо вектор v_1 до двійкової системи числення:

$$v_1^T = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$$

При першому кроці розв’язку підставимо значення вектора до формули АП враховуючи лише перші біти вектора:

$$1 + 0 - 3 \leq 2\ ;\ \ 0 \leq 4$$

4 в цьому випадку це результат обчислення формули при підставленні перших бітів, нехай $b_1 = 4$. Для наступного кроку прибираємо наймолодший біт цього значення і підставляємо як нове значення $b_1^* = 2$, з лівої сторони нерівності підставляємо значення других бітів вектора:

$$0 + 2 - 0 \leq 2\ ; 0 \leq 0$$

Вхідні дані закінчились, умова нерівності виконується, отже розв’язок прийнятий.

Далі розглянемо вектор v_2:

$$v_2^T = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$$

Так само виконуючи покрокове підставлення бітів вектора отримуємо:

$$1 + 0 - 0 \leq 2\ ;\ \ 0 \leq 1$$

На цьому кроці значення $b_1 = 1$. При побітовому зміщені відбувається втрата даних. Це буде важливо пізніше.

$$0 + 2 - 3 \leq 0;\ 0 \leq -1$$

Вхідні дані закінчились, умова нерівності не виконується, отже розв'язок відхилено.

З наведеного прикладу можемо зробити висновок – кожен з кроків такого побітового розв'язку формули змінює значення b_i, і після цієї зміни разом з бітовим здвигом, отримуючи значення b_i^* може бути використана в наступному кроці розв'язку формули.

Якщо говорити в термінах автоматів: на вхід автомата подаються вектор-символи, у яких координати мають значення бітів у відповідних розрядах координат вектора. Станами автомата виступатимуть значення b_i^*, якщо це значення рівне або перевищує нуль – то цей стан є допустимим. Для побудови автомата необхідно прорахувати всі можливі варіанти станів, тобто значень b_i^*, та переходів цих станів по всій множині вхідних символів. Наведемо більш формальний опис алгоритму:

Нехай дана довільна формула АП $(a, x) \leq b$, і порівнюється деякий вектор v. Позначатимемо координати вектора v до біту i як c^i. Перед прочитанням вхідного вектор-символу v_i знаходився у стані автомата $b_i{}'$, а після прочитання – в стані $b_{i+1}{}'$. Можемо визначити формулу цих станів автомата, згідно з роз'ясненнями, наведених вище:

$$b_i' = \left\lfloor \frac{1}{2^i}(b - (a, c^i)) \right\rfloor \; ; \; b_{i+1}' = \left\lfloor \frac{1}{2^{i+1}}(b - (a, c^{i+1})) \right\rfloor$$

Також, за наведеним вище означенням, визначимо формулу для c^{i+1}:

$$c^{i+1} = c^i + 2^i v_i$$

Підставимо цей вираз у попередню формулу:

$$b_{i+1}' = \left\lfloor \frac{1}{2^{i+1}}\left(b - (a, c^i) - 2^i(a, v_i)\right) \right\rfloor$$

$$2b_{i+1}' = \left\lfloor \frac{1}{2^i}\left(b - (a, c^i)\right) - (a, v_i) \right\rfloor$$

Підставляючи значення b_i' отримуємо фінальну формулу:

$$b_{i+1}' = \left\lfloor \frac{1}{2}(b_i' - (a, v_i)) \right\rfloor$$

Що збігається з побітовим методом вирішення форуми АП.

З цієї формули можемо визначити функцію переходів, та знайти всі можливі стани автомата. Початковим станом буде значення b формули АП. Для кожного стану визначаються всі можливі переходи в інші стани. Якщо перехід приводить до стану, що ще не існує, такий стан створюється і для нього також визначаються всі можливі переходи.

Наведемо приклад побудови автомата для формули АП $x_1 - 2x_2 \leq 2$:

Допустимі символи автомата: $z_1 = \binom{0}{0}, z_2 = \binom{0}{1}, z_3 = \binom{1}{0}, z_4 = \binom{1}{1}$

Початковий стан: $b = 2$

$$b_1' = \left\lfloor \frac{1}{2}(2 - (0 - 0)) \right\rfloor = 1\,, b_2' = \left\lfloor \frac{1}{2}(2 - (0 - 2)) \right\rfloor = 2,$$

$$b_3' = \left\lfloor \frac{1}{2}(2 - (1 - 0)) \right\rfloor = 0\,, b_4' = \left\lfloor \frac{1}{2}(2 - (1 - 2)) \right\rfloor = 1$$

Отримуємо два нових стани, знаходимо перехід кожного з них: для $b = 0$:

$$b_1' = \left\lfloor \frac{1}{2}(0 - (0 - 0)) \right\rfloor = 0\,, b_2' = \left\lfloor \frac{1}{2}(0 - (0 - 2)) \right\rfloor = 1,$$

$$b_3' = \left\lfloor \frac{1}{2}(0 - (1 - 0)) \right\rfloor = -1\,, b_4' = \left\lfloor \frac{1}{2}(0 - (1 - 2)) \right\rfloor = 0$$

Для $b = 1$:

$$b_1' = \left\lfloor \frac{1}{2}(1 - (0 - 0)) \right\rfloor = 0\,, b_2' = \left\lfloor \frac{1}{2}(1 - (0 - 2)) \right\rfloor = 1,$$

$$b_3' = \left\lfloor \frac{1}{2}(1 - (1 - 0)) \right\rfloor = 0\,, b_4' = \left\lfloor \frac{1}{2}(1 - (1 - 2)) \right\rfloor = 1$$

Для $b = -1$:

$$b_1' = \left\lfloor \frac{1}{2}(-1 - (0 - 0)) \right\rfloor = -1\,, b_2' = \left\lfloor \frac{1}{2}(-1 - (0 - 2)) \right\rfloor = 0,$$

$$b_3' = \left\lfloor \frac{1}{2}(-1 - (1 - 0)) \right\rfloor = -1\,, b_4' = \left\lfloor \frac{1}{2}(-1 - (1 - 2)) \right\rfloor = 0$$

Отже, маємо наступний набір переходів:

$$2 \xrightarrow{z_2} 2, 2 \xrightarrow{z_1, z_4} 1, 2 \xrightarrow{z_3} 0\,; \quad 1 \xrightarrow{z_2, z_4} 1, 1 \xrightarrow{z_1, z_3} 0\,;$$

$$0 \xrightarrow{z_2} 1, 0 \xrightarrow{z_1, z_4} 0, 0 \xrightarrow{z_3} -1\,; \; -1 \xrightarrow{z_2, z_4} 0, -1 \xrightarrow{z_1, z_3} -1$$

Графічне зображення автомата, створеного за вищезазначеним набором переходів подано на рисунку 2.3

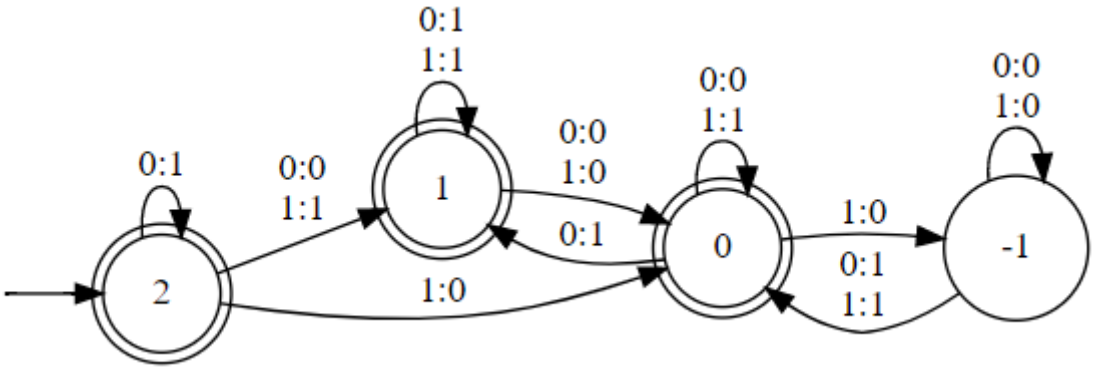


Рис 2.3

Також дамо оцінку максимальної кількості станів автомата, що будується за даним алгоритмом. Для цього визначимо межі можливих значень станів автомата:

Нехай дано формулу АП $(a, c) \leq b$, $m = \sum_{i=1}^{q} |a_i|$, тоді стани j що породжуються при побудові автомата за алгоритмом задовольняють нерівності $-|b| - m \leq j \leq |b| + m$.

Доведення: умова виконується для початкового стану b. Припустимо що умова виконується для довільного стану b_i, доведемо що вона також виконується для наступного стану b_{i+1}, використовуючи формулу наступного стану $b_{i+1} = \left\lfloor \frac{1}{2}(b_i - (a, z)) \right\rfloor$:

$$-|b| - m \leq b_{i+1} \leq |b| + m$$

$$\left\lfloor \frac{-|b| - m - (a, z)}{2} \right\rfloor \leq b_i \leq \left\lfloor \frac{|b| + m - (a, z)}{2} \right\rfloor$$

Оскільки $-|b| - m \leq \frac{-|b|-2m}{2} \leq \left\lfloor \frac{-|b|-m-(a,z)}{2} \right\rfloor$ та $\left\lfloor \frac{|b|+m-(a,z)}{2} \right\rfloor \leq \frac{|b|+2m}{2} \leq |b| + m$ робимо висновок що $-|b| - m \leq b_{i+1} \leq |b| + m$ також виконується. Враховуючи що твердження виконується для початкового стану b виходить що припущення для довільного стану b_i також виконується.

Згідно з отриманими межами значень станів можемо дати оцінку їх кількості:

$$|Q| \leq 2(|b| + m)$$

2.2.2.2 Визначення рівності у випадку невід'ємних чисел

Наступним кроком стане розгляд алгоритму побудови автомата, що приймає розв'язки формули АП вигляду $(a, x) = b$. Глянувши на автомат що було побудовано для розв'язку нерівності може здаватись що потрібно лише зробити всі стани орім нульового не допустимими, проте потрібно врахувати ще одну умову: при переходах між станами не може бути втрати даних при побітовому зміщені значення b_i', оскільки у випадку заокруглення вниз значення $b_i' \geq 0$ виконується, а при умові рівності заокруглення означає що рівність нулю неможлива.

Отже, для побудови автомата, що розв'язує формули АП вигляду $(a, x) = b$ необхідно дотримуватись тих самих правил що і при нерівності, за виключенням двох умов:

— Якщо при обрахунку значення b_{i+1}' до заокруглення вниз значення не є цілим – перехід не будується і вважається невизначеним

— Серед допустимих станів буде лише один – той, значення b_i' якого рівне нулю

Наведемо приклад побудови автомата для формули АП $x_1 + 2x_2 - 3x_3 = 2$:

Допустимі символи автомата:

$$z_1 = \begin{pmatrix}0\\0\\0\end{pmatrix}, z_2 = \begin{pmatrix}0\\0\\1\end{pmatrix}, z_3 = \begin{pmatrix}0\\1\\0\end{pmatrix}, z_4 = \begin{pmatrix}0\\1\\1\end{pmatrix},$$

$$z_5 = \begin{pmatrix}1\\0\\0\end{pmatrix}, z_6 = \begin{pmatrix}1\\0\\1\end{pmatrix}, z_7 = \begin{pmatrix}1\\1\\0\end{pmatrix}, z_8 = \begin{pmatrix}1\\1\\1\end{pmatrix}$$

Початковий стан $b = 2$

$$b_1' = \frac{1}{2}(2 - (0 + 0 - 0)) = 1\,, b_3' = \frac{1}{2}(2 - (0 + 2 - 0)) = 0$$

$$b_6' = \frac{1}{2}(2 - (1 + 0 - 3)) = 0, b_8' = \frac{1}{2}(2 - (1 + 2 - 3)) = 1$$

Переходи за вектор-символами z_2, z_4, z_5, z_7 відсутні, оскільки дають при діленні на два не цілі числа.

Для $b = 1$:

$$b_2' = \frac{1}{2}(1 - (0 + 0 - 3)) = 2\,,\, b_4' = \frac{1}{2}(1 - (0 + 2 - 3)) = 1$$

$$b_5' = \frac{1}{2}\big(1 - (1 + 0 - 0)\big) = 0, b_7' = \frac{1}{2}\big(1 - (1 + 2 - 0)\big) = -1$$

Для $b = 0$:

$$b_1' = \frac{1}{2}\big(0 - (0 + 0 - 0)\big) = 0\,,\, b_3' = \frac{1}{2}\big(0 - (0 + 2 - 0)\big) = -1$$

$$b_6' = \frac{1}{2}(0 - (1 + 0 - 3)) = 1, b_8' = \frac{1}{2}(0 - (1 + 2 - 3)) = 0$$

Для $b = -1$:

$$b_2' = \frac{1}{2}(-1 - (0 + 0 - 3)) = 1\,,\, b_4' = \frac{1}{2}(-1 - (0 + 2 - 3)) = 0$$

$$b_5' = \frac{1}{2}\big(-1 - (1 + 0 - 0)\big) = -1, b_7' = \frac{1}{2}\big(-1 - (1 + 2 - 0)\big) = -2$$

Для $b = -2$:

$$b_1' = \frac{1}{2}\big(-2 - (0 + 0 - 0)\big) = -1\,,\, b_3' = \frac{1}{2}\big(-2 - (0 + 2 - 0)\big) = -2$$

$$b_6' = \frac{1}{2}\big(-2 - (1 + 0 - 3)\big) = 0, b_8' = \frac{1}{2}\big(-2 - (1 + 2 - 3)\big) = -1$$

Отже, маємо наступний набір переходів:

$$2 \xrightarrow{z_1, z_8} 1, 2 \xrightarrow{z_3, z_6} 0\,; \quad 1 \xrightarrow{z_2} 2, 1 \xrightarrow{z_4} 1, 1 \xrightarrow{z_5} 0, 1 \xrightarrow{z_7} -1\,;$$

$$0 \xrightarrow{z_6} 1, 0 \xrightarrow{z_1, z_8} 0, 0 \xrightarrow{z_3} -1\,;\; -1 \xrightarrow{z_2} 1, -1 \xrightarrow{z_4} 0, -1 \xrightarrow{z_5} -1, -1 \xrightarrow{z_7} -2\,;$$

$$-2 \xrightarrow{z_6} 0, -2 \xrightarrow{z_1, z_8} -1, -2 \xrightarrow{z_3} -2$$

Графічне зображення автомата, створеного за вищезазначеним набором переходів подано на рисунку 2.4

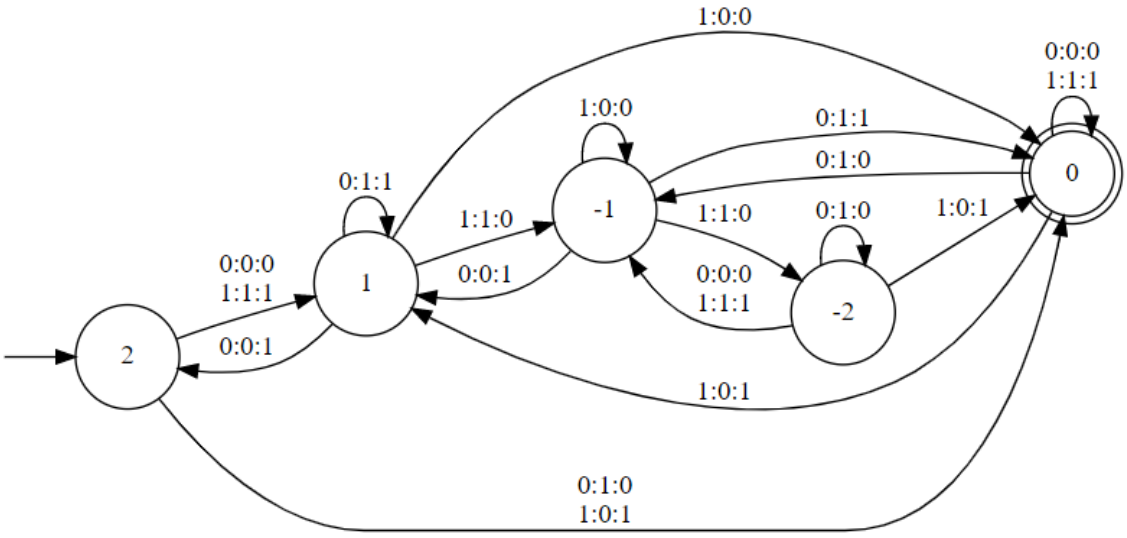


Рис 2.4

2.2.2.3 Визначення нерівності у випадку цілих чисел

Як і у випадку з розбиттям векторів відносно порогового вектора необхідно мати можливість представляти від'ємні значення координат векторів у вигляді 2-доповнення, і модифікувати наявні алгоритми для врахування цієї зміни представлення.

Для формул АП представлення координат у двійковому вигляді відбувається у порядку зростання розряду, а отже знаковий біт знаходитиметься наприкінці 2-доповнення. З цього визначаємо 2-доповнення для формул АП:

Для $x \geq 0$ двійкове зображення має вигляд $(x_0 x_1 \dots x_{n-1} x_n)$, де $x_n = 0$; для випадку для зображення числа $-x$ за двійковим зображенням числа x 2-доповнення визначається як

$$-x = \left(\sum_{i=0}^{n-1} \overline{x_i} 2^i - \overline{x_n} 2^n \right) + 1$$

де $\overline{x_n} = 1$,

$\overline{x_i} = 1 - x_i$ для $i = \overline{0, n-1}$

Наприклад слово 110 зображує число 1+2-0=3, слово 111 – 1+2-4=-1, 10011 – 1+0+0+8-16=-7.

Варто зазначити що таке 2-доповнення не є однозначним. Наприклад всі слова 1, 11, 111,… зображують одне й те саме число -1. З чого можемо припустити що регулярний вираз вигляду $x_0x_1 \dots x_{n-1}x_n(x_n)^*$ зображує одне й те саме число. Це твердження випливає з того факту що

$$-2^{m+n} + 2^{m-1+n} + \cdots + 2^{n+1} = 2^n$$

Приведемо приклад зображення вектора у вигляді 2-доповнення:

$$a^T = \begin{pmatrix} 12 \\ -6 \\ -15 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} = x_1x_2x_3x_4x_5(x_5)^*$$

$$x_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, x_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, x_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, x_4 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, x_5 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Про модифікацію наявних алгоритмів. Згідно з описом 2-доповнення до двійкового представлення числа, при покроковому розв’язані формули може з’являтись значення знакового біту, що фактично представляє цей розряд зі знаком мінус, що змінює знак формули переходів, що створює два переходи з одного стану (один перехід для звичного кроку розв’язку, інший перехід – для останнього кроку знакових бітів), тим самим створюючи недетермінований автомат.

Використовуючи алгоритми детермінізації автоматів можна уникнути потребу використання недетермінованого автомата.

В результаті алгоритм побудови автомата мало чим відрізняється від алгоритму побудови автомата для розв’язку формули АП лише з цілими числами, єдина відмінність полягає в відсутності заключних станів окрім одного, в який можливо переходити тільки на останньому кроці виконання автомата.

Опишемо алгоритм побудови недетермінованого автомата псевдокодом:

НДСА-Нерівність

Вхід: коефіцієнти $a = (a_1, \dots, a_n)$ формули АП $(a, x) \leq b$

Вихід: НДСА $A = (A, X, f, a_0, F)$, що приймає розв’язки формули АП

$A, f, F \coloneqq \emptyset;\ a_0 = \{b\};$

$W = \{b\};$

поки $W \neq \emptyset$:

взяти s з W

додати s до A

для кожного z із множини $\{0,1\}^n$:

$j \coloneqq \left\lfloor \frac{1}{2}(s - (a,z)) \right\rfloor$

якщо $j \notin A, j \notin W$ тоді додати j до W

додати (s,z,j) до f

$c' \coloneqq s + a * z$

якщо $c' \geq 0$ тоді

якщо $a_f \notin A$ додати a_f до A та F

додати (s,z,a_f) до f

повернути $((A,X,f,a_0,F))$

Приведемо приклад побудови НДСА для формули АП $2x - y \leq -1$:

Допустимі символи автомата: $z_1 = \binom{0}{0}, z_2 = \binom{0}{1}, z_3 = \binom{1}{0}, z_4 = \binom{1}{1}$

Для $b = -1$:

$$b_1' = \left\lfloor \frac{1}{2}(-1 - (0-0)) \right\rfloor = -1\,, b_2' = \left\lfloor \frac{1}{2}(-1 - (0-1)) \right\rfloor = 0,$$

$$b_3' = \left\lfloor \frac{1}{2}(-1 - (2-0)) \right\rfloor = -2\,, b_4' = \left\lfloor \frac{1}{2}(-1 - (2-1)) \right\rfloor = -1$$

Перевірка умови $s + a * z \geq 0$, тобто $b + a * z \geq 0$:

$$\mathrm{c}_1' = \frac{1}{2}(-1 + (0-0)) = -0{,}5\,, \mathrm{c}_2' = \frac{1}{2}(-1 + (0-1)) = -1,$$

$$\mathrm{c}_3' = \frac{1}{2}(-1 + (2-0)) = 0{,}5\,, \mathrm{c}_4' = \frac{1}{2}(-1 + (2-1)) = 0$$

Виконана для вектор-символів z_3, z_4;

Для $b = 0$:

$$b_1' = \left\lfloor \frac{1}{2}(0 - (0-0)) \right\rfloor = 0\,, b_2' = \left\lfloor \frac{1}{2}(0 - (0-1)) \right\rfloor = 0,$$

$$b'_3 = \left\lfloor \frac{1}{2}(0-(2-0)) \right\rfloor = -1\,, b'_4 = \left\lfloor \frac{1}{2}\big(0-(2-1)\big) \right\rfloor = -1$$

$$\mathrm{c}'_1 = \frac{1}{2}\big(0+(0-0)\big) = 0\,, \mathrm{c}'_2 = \frac{1}{2}\big(0+(0-1)\big) = -0{,}5,$$

$$\mathrm{c}'_3 = \frac{1}{2}\big(0+(2-0)\big) = 1\,, \mathrm{c}'_4 = \frac{1}{2}\big(0+(2-1)\big) = 0{,}5$$

Перевірка умови виконана для вектор-символів z_1, z_3, z_4;

Для $b = -2$:

$$b'_1 = \left\lfloor \frac{1}{2}(-2-(0-0)) \right\rfloor = 0\,, b'_2 = \left\lfloor \frac{1}{2}(-2-(0-1)) \right\rfloor = 0,$$

$$b'_3 = \left\lfloor \frac{1}{2}(-2-(2-0)) \right\rfloor = -1\,, b'_4 = \left\lfloor \frac{1}{2}\big(-2-(2-1)\big) \right\rfloor = -1$$

$$\mathrm{c}'_1 = \frac{1}{2}\big(-2+(0-0)\big) = -1\,, \mathrm{c}'_2 = \frac{1}{2}\big(-2+(0-1)\big) = -1{,}5,$$

$$\mathrm{c}'_3 = \frac{1}{2}\big(-2+(2-0)\big) = 0\,, \mathrm{c}'_4 = \frac{1}{2}\big(-2+(2-1)\big) = -0{,}5$$

Перевірка умови виконана для вектор-символа z_3;

Отже, маємо наступний набір переходів:

$$0 \xrightarrow{z_1,z_2} 0, 0 \xrightarrow{z_3,z_4} -1\,;\ \ -1 \xrightarrow{z_2} 0, -1 \xrightarrow{z_1,z_4} -1, -1 \xrightarrow{z_3} -2\,;$$

$$-2 \xrightarrow{z_1 z_2} 0, -2 \xrightarrow{z_3,z_4} -1\,;$$

$$0 \xrightarrow{z_1,z_3,z_4} a_f, -1 \xrightarrow{z_3,z_4} a_f, -2 \xrightarrow{z_3} a_f$$

Графічне зображення автомата, створеного за вищезазначеним набором переходів подано на рисунку 2.5

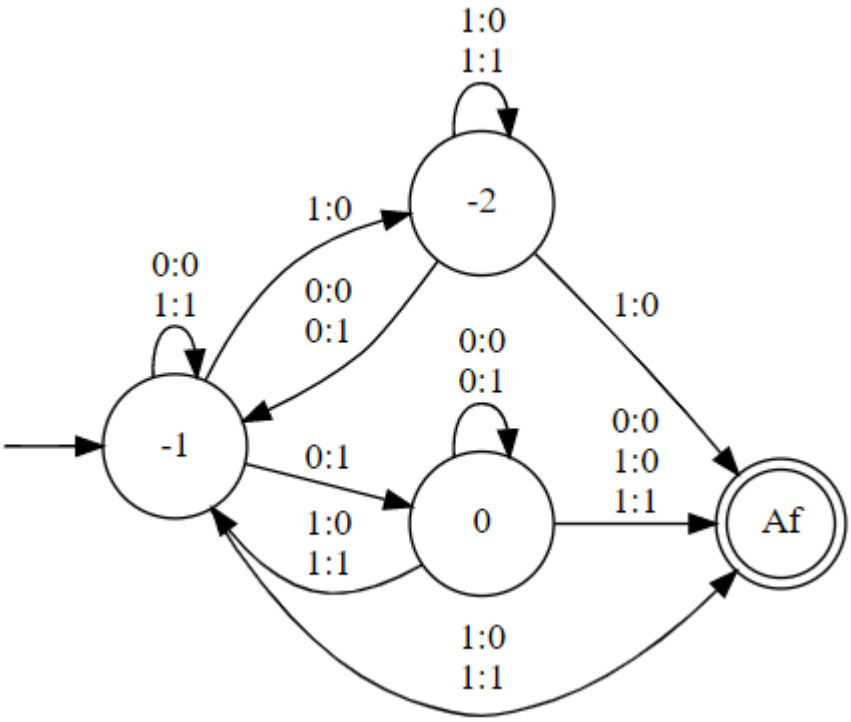


Рис 2.5

2.2.2.4 Визначення рівності у випадку цілих чисел

Аналогічно переходу з алгоритму для побудови автомата, що приймає розв'язки формули АП з нерівністю до алгоритму автомата, що приймає розв'язки формули АП з рівністю у множині цілих невід'ємних чисел, перехід до алгоритму з урахуванням від'ємних чисел потребує урахування двох умов: перехід в заключний стан a_f лише у випадку $s + a * z = 0$ та виключення з автомата переходів, що при обчисленні дають не цілі числа.

Звідси можемо описати алгоритм побудови автомата:

НДСА-Рівність

Вхід: коефіцієнти $a = (a_1, \dots, a_n)$ формули АП $(a, x) = b$

Вихід: НДСА $A = (A, X, f, a_0, F)$, що приймає розв'язки формули АП

$A, f, F \coloneqq \emptyset;\ a_0 = \{b\};$

$W = \{b\};$

поки $W \neq \emptyset$:

взяти s з W

додати s до A

для кожного z із множини $\{0,1\}^n$:

якщо $(s - az)\ mod\ 2 = 0$:

$j \coloneqq \frac{1}{2}\big(s - (a, z)\big)$

якщо $j \notin A, j \notin W$ тоді додати j до W

додати (s, z, j) до f

$c' \coloneqq s + a * z$

якщо $c' = 0$ тоді

якщо $a_f \notin A$ додати a_f до A та F

додати $\big(s, z, a_f\big)$ до f

повернути $\big((A, X, f, a_0, F)\big)$

Приведемо приклад побудови НДСА для формули АП $2x - y = -1$:

Допустимі символи автомата: $z_1 = \binom{0}{0}, z_2 = \binom{0}{1}, z_3 = \binom{1}{0}, z_4 = \binom{1}{1}$

Для $b = -1$:

$$b_2' = \frac{1}{2}\big(-1 - (0 - 1)\big) = 0, b_4' = \frac{1}{2}\big(-1 - (2 - 1)\big) = -1$$

Перевірка умови $s + a * z = 0$, тобто $b + a * z = 0$:

$$\mathrm{c}_1' = \frac{1}{2}\big(-1 + (0 - 0)\big) = -0{,}5\ , \mathrm{c}_2' = \frac{1}{2}\big(-1 + (0 - 1)\big) = -1,$$

$$\mathrm{c}_3' = \frac{1}{2}(-1 + (2 - 0)) = 0{,}5\ , \mathrm{c}_4' = \frac{1}{2}\big(-1 + (2 - 1)\big) = 0$$

Виконана для вектор-символа z_4;

Для $b = 0$:

$$b_1' = \frac{1}{2}\big(0 - (0 - 0)\big) = 0, b_3' = \frac{1}{2}(0 - (2 - 0)) = -1$$

$$\mathrm{c}_1' = \frac{1}{2}\big(0 + (0 - 0)\big) = 0\ , \mathrm{c}_2' = \frac{1}{2}\big(0 + (0 - 1)\big) = -0{,}5,$$

$$c'_3 = \frac{1}{2}\big(0 + (2 - 0)\big) = 1\,,\, c'_4 = \frac{1}{2}\big(0 + (2 - 1)\big) = 0{,}5$$

Перевірка умови виконана для вектор-символів z_1;

Графічне зображення автомата, створеного за вищезазначеним набором переходів подано на рисунку 2.6

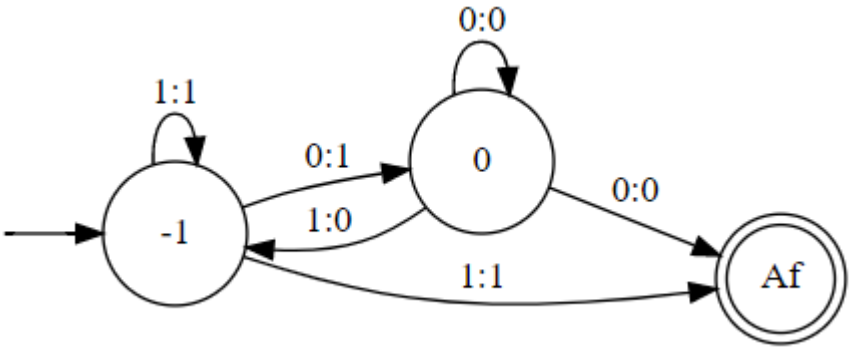


Рис 2.6

2.2.3 Створення алгоритму розбиття відносно гіперплощини

Розгляд алгоритмів побудови автоматів для розв'язку формул АП дає можливість створити уявлення про вимоги до автомата що виконує розподіл векторів відносно гіперплощини, заданої у вигляді формули АП

Автомат що виконує розбиття має визначати один з трьох відношень вектора до гіперплощини: лежить на площині, над площиною або під площиною. Проте до цього часу алгоритми визначали лише одне обмеження розв'язку відносно формули. Тому необхідні зміни до алгоритму побудови автомата.

Найбільша проблема раніше наведених алгоритмів – втрата точності в результаті заокруглення. Для її вирішення вводяться додаткові стани – стани, що відповідають проміжному результату обрахування формули, але враховують факт втрати даних – тобто побітового здвигу що втратив одиницю. Оскільки повернення до відповідного розряду відбуватись не може, то і перехід до стану з міткою про втрату даних відбуватиметься лише один раз. Зі стану з міткою можливі переходи лише до інших станів з міткою або до допустимих станів нерівності. Зі стану без мітки можливі переходи до інших станів без мітки, станів з міткою, допустимих станів рівності та нерівності.

Враховуючи вищесказане, можемо описати алгоритм побудови автомата що виконує розбиття векторів відносно гіперплощини:

НДСА-Розбиття

Вхід: коефіцієнти $a = (a_1, \dots, a_n)$ формули АП $(a, x) = b$

Вихід: НДСА $A = (A, X, f, a_0, F)$, що визначає відношення вектора до гіперплощини

$A, f, F \coloneqq \emptyset;\ a_0 = \{b\};$

$W = \{b\};$

поки $W \neq \emptyset$:

взяти s з W

додати s до A

для кожного z із множини $\{0,1\}^n$:

якщо $(s - az)\ mod\ 2 = 0$ та БЕЗ_МІТКИ(s):

$j \coloneqq \frac{1}{2}\big(s - (a, z)\big)$

якщо $j \notin A, j \notin W$ тоді додати j до W

додати (s, z, j) до f

$c' \coloneqq s + a * z$

якщо $c' = 0$ тоді:

якщо $a_= \notin A$ додати $a_=$ до A та F

додати $(s, z, a_=)$ до f

якщо $c' > 0$ тоді:

якщо $a_> \notin A$ додати $a_>$ до A та F

додати $(s, z, a_>)$ до f

якщо $(s - az)\ mod\ 2 \neq\ 0$ або ¬БЕЗ_МІТКИ(s):

$j \coloneqq \left\lfloor \frac{1}{2}\big(s - (a, z)\big) \right\rfloor$

якщо $j \notin A, j \notin W$ тоді додати j до W

додати (s, z, j) до f

$c' \coloneqq s + a * z$

якщо $c' \geq 0$ тоді:

якщо $a_> \notin A$ додати $a_>$ до A та F

додати $(s, z, a_>)$ до f

В алгоритмі приймаються вектори, що лежать на гіперплощині або над площиною. У випадку коли вектор не прийнятий автоматом – вектор лежить під площиною.

Графічне зображення автомата, створеного програмною системою для розбиття векторів відносно гіперплощини $2x - y = -1$ подано на рисунку 2.7

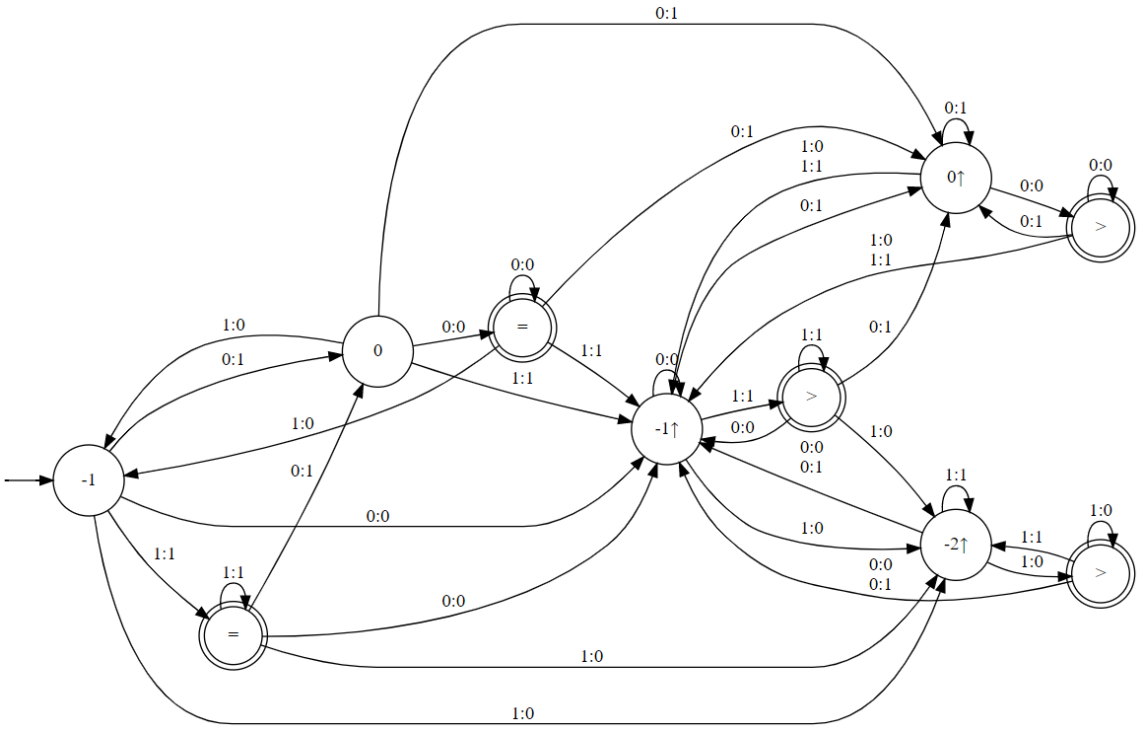


Рис 2.7

Для всіх попередніх алгоритмів оцінка кількості станів результуючого автомату була однаковою. У даному алгоритмі для кожного зі станів може існувати парний стан з міткою, з чого випливає оцінка кількості станів автомата для розбиття векторів відносно гіперплощини що задана формулою $a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$:

$$|Q| = 4(|b| + m)$$

де $m = \sum_{i=1}^{q} |a_i|$

РОЗДІЛ 3 ОПИС ПРОГРАМНОЇ СИСТЕМИ

3.1 Призначення програмної системи

Створена програмна система може використовуватись для вивчення алгоритмів побудови автоматів, наочного ілюстрування процесу розв’язку задач заданих для виконання автоматним методом та власне розв’язок задач з використанням автоматного методу.

3.2 Опис реалізація програмної системи

Програмна система створена у вигляді застосунку з інтерактивним вікном користувацького інтерфейсу. Користувацький інтерфейс створений засобами бібліотеки Qt. Приклад вигляду вікна користувацького інтерфейсу подано у додатку Б.

3.2.1 Віджет основного вікна

Віджет основного вікна містить в собі контейнери для відображення віджета вводу даних для побудови автоматів та векторів для порівняння відносно гіперплощини або порогового вектора, віджета результатів розбиття векторів, віджета стану розв’язку конкретної задачі. Також окремо міститься віджет, що має можливість виводити зображення, подані у форматі SVG та кнопки, що відповідають за швидкість зміни станів автоматів, що зображуються на віджеті зображення.

3.2.2 Віджети вводу даних

Віджети вводу даних містяться у контейнері основного вікна у вигляді вкладок. У програмній системі наявні дві вкладки вводу – одна відповідає за ввід даних для порівняння множини векторів з одним вказаним пороговим вектором, інша – за ввід даних для порівняння множини векторів відносно вказаної

гіперплощини. Ці віджети мають деякі кнопки для керування процесом вводу даних – підтвердження вводу порогового вектора або формули гіперплощини, додання вектора до множини векторів, що порівнюється. Внизу віджетів – кнопки скиду введених даних, заповнення даних випадково згенерованими, кнопка зупинення та відновлення роботи активного автомата та кнопка початку роботи автомата після введення порогового вектора або формули гіперплощини та хоча б одного вектора до множини векторів, що порівнюються.

Також у коді реалізації цих віджетів наявні методи для початку роботи автомата, і об’єкт класу, що керує процесами створення, виведення на користувацьке вікно, обробку вхідних символів та передачею їх до автомата, оновленням зображення автомата та виводу проміжних та фінальних результатів роботи автомата.

3.2.3 Віджети результатів розбиття множини векторів

Віджети результатів розбиття множини векторів містять в собі контейнери для векторів, що пройшли розбиття, і потрапляють до одного з них. В програмній системі наявні два види таких віджетів: з чотирма контейнерами, що відповідають порядку відношень між векторами $\{=, >, <, \propto\}$, використовується при розбитті множини векторів відносно порогового вектора; та з трьома контейнерами, що відповідають відношенням $\{=, >, <\}$, використовується при розбитті множини векторів відносно гіперплощини.

3.2.4 Віджети стану розв’язку

Віджети стану розв’язку виводять у вікні програми деякі відомості про стан розв’язку розбиття окремого вектора відносно порогового або гіперплощини, серед даних що виводяться наявні: представлення об’єктів, що порівнюються; представлення залишку координат у двійковому вигляді, що ще не були оброблені

автоматом; проміжні результати порівняння, у випадку порівняння відносно порогового вектора – результат вже порівняних координат векторів.

3.2.5 Віджет зображення SVG

Віджет QSvgWidget – елемент користувацького інтерфейсу, що надається бібліотекою Qt для відображення зображень у форматі SVG. У програмному коді надається можливість використовувати метод load об’єкту віджета, з масивом байт як аргумент методу, що містить зображення у форматі SVG.

Є можливість створювати jpeg, png, або деякі інші формати зображень з використанням пакета інструментів Graphviz, проте для виділення одного стану я поточного необхідно повторно викликати один з інструментів Graphviz, а один виклик, навіть при зображенні малих графів, займає від третини секунди, що доволі багато часу для однопоточної програми з графічним користувацьким інтерфейсом, оскільки може здаватись що програма просто «висне». Тому в програмній системі зображення представляються у форматі SVG. Цей формат є підмножиною мови розмітки XML. Мова програмування Python надає однойменний модуль xml для роботи з даними у форматі XML, що дозволяє швидко редагувати деякі частини зображення, тим самим виключаючи проблему затримки під час роботи автомата.

3.2.6 Програмна реалізація автомата

Автомати у програмній системі реалізовані за принципом ООП – наявні класи об’єктів в цілому для автомата і клас об’єктів для станів автомата. Об’єкт автомата зберігає в собі поточний стан під час виконання та керує переходами через наявні переходи поточного стану в залежності від вхідних символів, що надходять до автомата. Об’єкти станів містять в собі дані про стан, такі речі як допустимість стану та результат, що представляє цей стан, а також переходи цього стану до інших.

3.2.7 Швидкодія розбиття відносно гіперплощини

Для розуміння швидкодії системи була створена функція, що вимірює час обрахування розбиття використовуючи наївний метод – просто підставлення координат вектора, що розбивається, у формулу гіперплощини, що розбиває вектор. Далі будується автомат для розбиття відносно цієї площини й цей самий вектор розбивається відносно площини використовуючи цей автомат.

Коефіцієнти формули гіперплощини та координати вектора генеруються випадково, з обмеженням на модуль значення коефіцієнтів та координат $n, |a_i| < n;\ |v_i| < n, i = \overline{1, l}$ та вказаною кількістю коефіцієнтів l.

Таблиця 3.1 – виміри швидкодії системи

#	n	l	Наївний метод	Час побудови автомату	Час розв'язку	К-сть станів
1	10	3	1255нс	1.31мс	6359нс	61
2	20	3	1277нс	3.21мс	7427нс	118
3	40	3	1707нс	8.31мс	8974нс	224
4	10	5	1844нс	13.46мс	10390нс	368
5	20	5	2599нс	37.91мс	11553нс	738
6	10	8	2659нс	274.43мс	12623нс	5003
7	10	10	3146нс	1.71с	11115нс	24037
8	7	12	3624нс	5.78с	10939нс	84928
9	5	15	4247нс	47.09с	12547нс	638230

З таблиці 3.1 бачимо що наївний метод має велику залежність часу виконання від кількості коефіцієнтів l, що є очікуваним результатом. Час розв'язку з використанням автомата має більший початковий час, проте залежність часу виконання від l наявна лише через метод реалізації, і більшій мірі залежіть від довжини двійкового представлення координат вектора що розбивається, тобто від n. При достатньо великому l автоматний метод випередить наївний метод, проте через велику кількість станів, яка значно росте та залежить від l та n та через реалізацію системи методом ООП створене представлення автомата займає багато оперативної

пам’яті, тому на практиці це твердження перевірити неможливо. В цілому система показує потенціал, і враховуючи відсутність пріоритету швидкодії в створеній програмній системі можемо зробити висновок про достатню ефективність розробленого алгоритму.

ВИСНОВКИ

В процесі виконання роботи були проведені дослідження алгоритмів побудови автоматів, що розв’язують задачі розбиття множини векторів відносно порогового вектора та автоматів, що розв’язують задачі перевірки розв’язків формули арифметики Пресбургера. На основі набутих знань було розроблено алгоритм для створення автомата, що розв’язує задачу розбиття множини векторів відносно заданої гіперплощини.

Також отримані знання були використані на практиці – була створена програмна система, що реалізує досліджені та розроблені алгоритми створення автоматів для виконання задач розподілу, система може використовувати створені автомати для розв’язку задачі розподілу множини векторів відносно порогового вектора, та відображає процес розв’язку задачі візуалізацією на вікні користувацького інтерфейсу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. McCulloch W. S. A logical calculus of the ideas immanent in nervous activity / W.S. McCulloch, W. Pitts // Bulletin of Mathematical Biophysics. – 1943
2. Mealy G. A Method to Synthesizing Sequential Circuits / George H. Mealy // Bell System Technical Journal / George H. Mealy., 1955. – С. 1045–1079.
3. Moore E. Gedanken-experiments on Sequential Machines / Edward F. Moore // Automata Studies, Annals of Mathematical Studies / Edward F. Moore., 1956. – С. 129–153.
4. Boudet A. Diophantine equations, Presburger arithmetic and finite automata / A. Boudet, H. Comon // Colloquium on Trees in Algebra and Programming / A. Boudet, H. Comon., 1996. – С. 30–43.
5. Lazar G. Mastering Qt 5: Create stunning cross-platform applications using C++ with Qt Widgets and QML with Qt Quick / G. Lazar, R. Penea., 2018. – 534 с. – (2)
6. Extensible Markup Language (XML) [Електронний ресурс] // 5. – 2008. – Режим доступу до ресурсу: https://www.w3.org/TR/xml/
7. PyGraphviz Documentation [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://pygraphviz.github.io/documentation/stable/pygraphviz.pdf
8. GANSNER E. An open graph visualization system and its applications to software engineering [Електронний ресурс] / E. GANSNER, S. NORTH. – 1999. – Режим доступу до ресурсу: https://graphviz.org/documentation/GN99.pdf
9. Hagberg A. NetworkX Reference [Електронний ресурс] / A. Hagberg, D. Schult, P. Swart. – 2022. – Режим доступу до ресурсу: https://networkx.org/documentation/stable/_downloads/networkx_reference.pdf
10. Кривий С. Л. Скінченні автомати: теорія, алгоритми, складність / Сергій Лук'яненко Кривий., 2020. – 428 с.

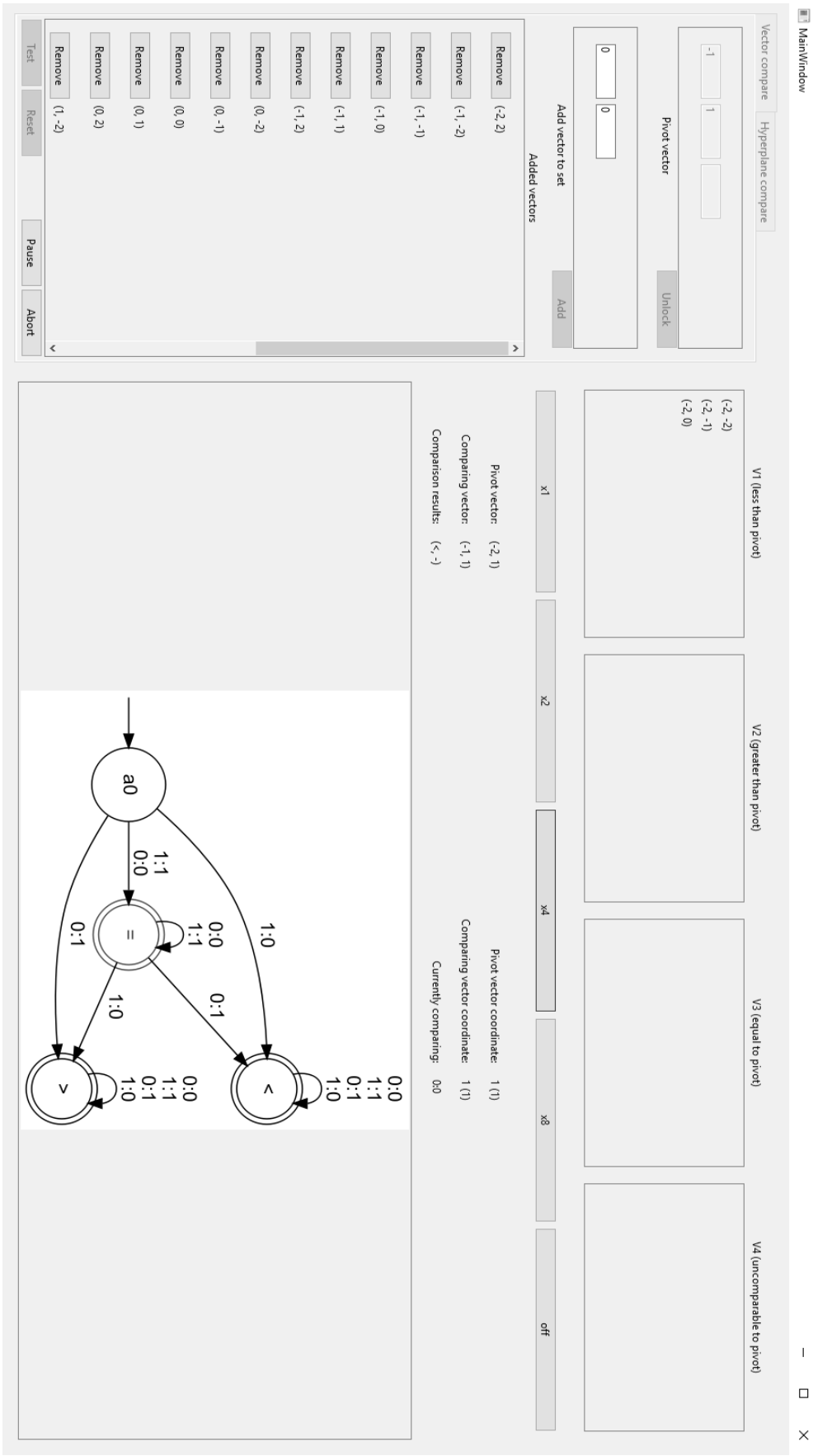
ДОДАТОК А

```
digraph G {
    rankdir=LR

    emnode [label="", shape="none", height=.0, width=.0]
    node1 [label="-1", shape="circle", width=.75]
    node3 [label="0", shape="circle", width=.75]
    node4 [label="Af", shape="doublecircle", width=.5]

    emnode -> node1
    node1 -> node3 [label=" 0:1 ", size=3]
    node1 -> node1 [label=" 1:1 ", size=3]
    node1 -> node4 [label=" 1:1 ", size=3]
    node3 -> node3 [label=" 0:0 ", size=3]
    node3 -> node1 [label=" 1:0 ", size=3]
    node3 -> node4 [label=" 0:0 ", size=3]
}
```

ДОДАТОК Б


MainWindow
Vector compare
Hyperplane compare
Pivot vector
Unlock
Add vector to set
Add
Added vectors
Remove (-2, 2)
Remove (-1, -2)
Remove (-1, -1)
Remove (-1, 0)
Remove (-1, 1)
Remove (-1, 2)
Remove (0, -2)
Remove (0, -1)
Remove (0, 0)
Remove (0, 1)
Remove (0, 2)
Remove (1, -2)
Test
Reset
Pause
Abort
V1 (less than pivot)
(-2, -2)
(-2, -1)
(-2, 0)
V2 (greater than pivot)
V3 (equal to pivot)
V4 (uncomparable to pivot)
x1
x2
x4
x8
off
Pivot vector: (-2, 1)
Comparing vector: (-1, 1)
Comparison results: (<, -)
Pivot vector coordinate: 1 (1)
Comparing vector coordinate: 1 (1)
Currently comparing: 0:0
a0
=
<
>
1:1
0:0
0:1
1:0