МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені

В.Н.Каразіна

Факультет математики і інформатики

Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота

бакалавр

на тему “Розв’язання деяких класів рівнянь у частково комутативних моноїдах”

Виконала: студентка 4 курсу, групи МФ-41

спеціальність 122 «Комп’ютерні науки»

освітньо-професійна програма «Інформатика»

Мисик О.Ф.

Керівник: к ф.-м. н., ст. викл. Полякова Л.Ю.

Харків - 2024

Зміст

1.Вступ

Роботу присвячено дослідженню деяких класів рівнянь у вільних частково комутативних моноїдах. Вільні частково комутативні моноїди було введено в 1969 році П'єром Картьє та Домініком Фоата [(1)] для комбінаторного доведення головної теореми МакМагона. Ця монографія була опублікована у Springer. Алгебраїчні та комбінаторні техніки, розроблені там, з тих пір використовуються в різних галузях математики, а також комп'ютерних наук. Зокрема в алгебрі згодом виникла теорія В'єннота [(1)] про купи шматків, яка стала дуже плідною в комбінаторній теорії ортогональних поліномів і в обчисленні твірних функцій багатьох змінних для поліномів. А в комп'ютерних науках Мазуркевичем [(2)] було запропоновано використовувати вільні частково комутативні моноїди (які в цьому контексті називають слідовими моноїдами) як математичну модель для дослідження комп'ютерного паралелізму. Використання слідових моноїдів в теорії паралельних обчислень зумовлено тим, що завдання, які можуть виконуватися незалежно одне від одного, можна подавати як комутуючі літери у слідах (тобто елементах слідових моноїдів), а некомутуючі -- як такі, що відповідають за блокування, точки синхронізації та об'єднання потоків.

Питання розв'язання буквених рівнянь у різноманітних алгебраїчних структурах набуло особливої значущості з часів 10-тої проблеми Гільберта, яка полягає у знаходженні способу, за допомогою якого можна встановити, чи має поліном з коефіцієнтами в цілих числах цілочисельний корінь, або ж ні Це одна з 23 задач, які Давід Гільберт запропонував у 1900 році на 2-ому Міжнародному конгресі математиків. Цілочисленне розв'язання алгебраїчних рівнянь цікавило математиків ще з античних часів. Наприклад, багато уваги приділялося рівнянню $x^2 + y^2 = z^2$. Евклід навів формули для знаходження усіх цілочисельних розв'язків цього рівняння. Деякі види рівнянь другого степеня та їх системи розглянув Діофант Александрійський, його роботи пізніше використовували Леонард Ейлер, П'єр Ферма, Жозеф Луі Лагранж, Карл Фрідріх Гаус та інші математики, які займалися теорією чисел. До 1768 року Лагранж повністю вивчив питання про цілочисельні розв'язки рівнянь другого степеня з двома невідомими.

Протягом 19 сторіччя багато математиків, шукаючи підходи до доведення Великої теореми Ферма, робили спроби дослідження діофантових рівнянь степеня вище за другий. Проте проблема розв’язання таких рівнянь залишалася актуальною: якогось загального методу отримано не було, знаходилися лише спеціальні прийоми для рівнянь певних типів. Швидше за все, Гільберт підозрював, що подальше дослідження цієї галузі призвело б до створення докладних і глибоких теорій, не обмежених діофантовими рівняннями, і це спонукало його внести завдання до списку для своєї доповіді. Доведення алгоритмічної нерозв'язності 10 проблеми було завершено в 1970.

Хоча питання щодо принципової алгоритмічної розв’язності рівнянь у частково комутативних моноїдах вже вирішено, цікавими залишаються задачі щодо побудови конкретних та ефективних алгоритмів для розв’язання рівнянь певного вигляду. Адже, оскільки сліди (тобто елементи слідових моноїдів) можуть моделювати послідовності певних дій у системі, враховуючи те, що виконання окремих кроків може здійснюватися залежно або незалежно один від одного, розв'язання рівнянь у слідових моноїдах дозволяє відповідати на питання щодо побудови систем із заданими властивостями. Зокрема розв'язання рівнянь вигляду $x^n=u$ дозволяє відповісти на питання, чи можна певну послідовність дій подати у вигляді повторень однієї й тієї самої послідовності елементарних кроків, тобто звести проектування системи до проектування меншої і простішої системи. Ці спостереження зумовлюють актуальність даної роботи.

Метою роботи є розгляд основних визначень з теорії частково комутативних моноїдів та деяких класів рівнянь у слідових моноїдах, пропонування техніки відновлення слідів за їхніми проекціями і застосовування цієї техніки до розв'язання рівнянь (зокрема, рівнянь вигляду $xu = vx$ та $x^n = u$), а також реалізація алгоритмів пошуку розв'язків. Структура роботи є наступною: у розділі 2 наведено огляд літератури, у розділі 3 викладено основні відомості з теорії вільних частково комутативних моноїдів, у розділі 4 поставлено задачу щодо розв'язання деяких класів рівнянь та розглянуто техніку відновлення слідів за їхніми проекціями і застосовано цю техніку до розв'язання рівнянь, у розділі 5

наведено деталі алгоритмічної реалізації запропонованих підходів до розв'язання рівнянь.

2. Огляд літератури

2.1. Паралелізм, комбінаторика та сліди

Паралелізм – фундаментальне поняття в інформатиці. Специфікація і верифікація паралельних програм мають першочергове значення. Для цього було запропоновано декілька формалізмів, як, наприклад, мережі Петрі, CSP та CCS Хоара та Мільнера. На основі поведінки елементарних мережевих систем Мазуркевич ввів концепцію часткової комутації. З одного боку, часткова комутація справляється з деякими важливими явищами в паралельності, а з іншого боку, вона близька до теорії вільних моноїдів, що описують послідовні програми. (3)

Одним із мотивів використання паралелізму є просто підвищення ефективності послідовної програми. За допомогою масивного паралелізму ми можемо сподіватися вирішити проблеми, які інакше були б нерозв'язними. Крім цього, паралелізм також служить як основа для загальної методології розробки або специфікації програмного забезпечення (і обладнання). (4)

Інший важливий аспект паралелізму полягає в тому, що багато існуючих і все більше число майбутніх систем за своєю суттю є паралельними. Широкий асортимент таких систем включає розподілені, децентралізовані, слабо або тісно пов'язані, вбудовані та реактивні системи. Проектування, аналіз і перевірка таких паралельних систем зазвичай виявляються дуже складними завданнями.

На це є, по суті, дві причини. Перша полягає в тому, що поведінка одночасних систем є набагато складнішою, ніж у аналогічних послідовних централізованих систем. Наявність паралельності означає, що в більшості випадків поведінку компонента не можна зрозуміти ізольовано. Треба завжди враховувати його численні взаємодії з іншими одночасно активними частинами системи. Друга причина стосується формальних методів. Вони є далекими від достатнього розвитку для суворої формальної обробки одночасної системи. Але без задовільного теоретичного обґрунтування та хорошої формальної основи аналіз і перевірка паралельних систем може базуватися лише на спеціальних доказах. Оскільки знайти спеціальні докази важко, це зазвичай дорого. Навіть гірше, часто це призводить до помилок. Зі зростанням важливості паралельності в інформатиці

така ситуація неприпустима. Тому нам потрібна глибша математична теорія паралелізму. (4) Мазуркевичем було відкрито, що поведінка безпечних мереж Петрі найкраще за все описується слідами. (див. також (4)). Застосування теорії слідів до мереж Петрі викликає постійний інтерес. Зокрема, була визнана синхронізація мов слідів як інструмент для модульного обчислення мережевих мов. Сліди – це абстрактна модель, що описує причинно-наслідкові зв'язки між виконаними діями також, наприклад, в EN-системах. Вони описують незалежність, отже, можливість виконуватися в довільному порядку (а також разом) для деяких дій. Структурно парні дії з наборами сусідніх місць, що не перетинаються, знаходяться у відношенні незалежності. Вільні частково комутативні моноїди (або слідові моноїди) служать основним алгебраїчним інструментом для дослідження паралельних систем. Представлені атомарні дії літерами, а самостійність дій відображається відношенням незалежності на алфавіті. Незалежність визначає часткову комутацію. Якщо кожна атомна дія a має зворотню $\overline{a}$, то на алгебраїчному рівні ми переходимо від моноїдів до групи. Це означає, що ми працюємо з вільними частково комутативними групами, які в алгебрі також відомі як групи графів. (5)

Теорія формальних мов над слідами почалась з вивчення розпізнаваних підмножин вільних частково комутативних моноїдів. Розпізнавані множини є фундаментальними для будь-якої теорії формальних мов. Вони мать ще більше значення, оскільки відповідають керованим структурам з кінцевим станом. Тому їм було приділено багато уваги; зокрема, було вивчено зв’язок з раціональними множинами і питання розв’язності. Основними досягненнями в цій галузі є опис Очманськи розпізнаваних мов і робота Зельонки з асинхронних автоматів. Великий інтерес представляє також теорія розв’язків рівнянь над частково комутативними змінними. (4)

Вільні частково комутативні моноїди і групи (також відомі як RAAG - прямокутні групи Артіна) також є об’єктом вивчення теоретичної інформатики. Питання розв’язності певних задач цілком природно виникають спочатку для

вільних моноїдів та груп, а потім для їхніх узагальнень – вільних частково комутативних моноїдів та груп. .

Вивчення рівнянь на множині слів (тобто у вільних моноїдах) є частиною комбінаторики слів уже понад півстоліття. З самого початку мотивація походила частково з теорії груп: метою було розуміти та параметризувати розв'язки рівнянь у вільних групах. Наприклад, Ліндону і Шютценбергеру потрібні були складні комбінаторні аргументи, щоб дати параметризований розв'язок рівняння $a^m = b^n c^p$ у вільній групі. З іншого боку, відомо, що параметричний опис множини розв'язків не завжди можливий. Проблема виконуваності (тобто проблеми, чи має дане рівняння розв'язки) рівняннь на множині слів у вільних моноїдах і вільних групах була основною проблемою завдяки її зв'язку з десятою проблемою Гільберта.

2.2. Квадратичні рівняння слідів з інволюцією

Хоча розв'язність проблеми виконуваності була доведена як для вільних моноїдів, так і для вільних частково комутативних, алгоритми розв'язання конкретних рівнянь все ще викликають інтерес, адже загальні є занадто складними. У цьому розділі ми коротко опишемо ситуацію з розв'язанням квадратичних рівнянь. Дослідження цих рівнянь дійшли до розгляду слідових моноїдів з інволюцією. Інволюція відповідає взяттю обернених елементів. У випадку груп інволюція зазвичай не має нерухомих точок на твірних , оскільки в вільному випадку немає нетривіальних елементів порядку два. Але у слідових моноїдах вважають, що твірні літери є нерухомими точками щодо інволюції. Алгоритм, який за лінійний час розв'язує систему квадратних рівнянь слів з інволюцією у слідових моноїдах описано у роботі (6). Основна ідея полягає в тому, щоб перетворити рівняння або систему рівнянь на іншу систему деякого стандартизованого вигляду (можливо, з більшою кількістю змінних, але з рівняннями простого вигляду), з подальшим виключенням змінних з цієї системи за певними правилами. Автори доводять збіжність процедури і зазначають, що отриманий алгоритм належить до класу PSPACE. Також в роботі розглядається алгоритм, який розв'язує квадратичні системи з додатковим обмеженням на можливу довжину розв'язків, модифікуючи

алгоритм з роботи (7) і показують, що проблема існування розв’язку в таких рівнянь є NP-важкою. (6)

2.3. Алгебраїчна структура комбінованих слідів

Окрім слідів у аналізі та верифікації паралельних систем використовуються їх розширення, яке називається комбінованими слідами (comtraces). Обидві моделі базуються на концепціях, що походять з теорії формальних мов, і вони здатні вловити поняття причинності та одночасності атомарних дій, які відбуваються під час процесу роботи системи. Перехід до області слідів і розвитку деяких фундаментальних понять, які виявилися успішними в теорії слідів є метою роботи «Algebraic structure of combined traces» Lukasz Mikulski (8)

Динамічна поведінка паралельних систем зазвичай описується як послідовності атомарних дії таких систем, що призводить до її формальної мовної семантики. Використовуючи цей простий підхід, ми не можемо виразити деякі явища, наприклад, одночасність і причинність, які є вирішальними у процесі розуміння та аналізу одночасної поведінки системи. У випадку конкретної операційної моделі можна розглянути можливість розширення послідовного опису шляхом додавання деякої інформації про відповідні властивості поведінки. Це можна зробити за допомогою розглядання послідовності кроків дій і додавання деяких причинно-наслідкових залежностей між діями. Саме цей підхід реалізовано у роботі (8) Автор досліджує алгебраїчну внутрішню структуру комбінованих слідів. Він починає зі згадування деяких стандартних понять про формальні мови, сліди та комбіновані сліди. Зокрема, дано визначення лексикографічного порядку на крокових послідовностях. Також розглядається канонічна форма Фоати комбінованого сліду, що виявляється максимальною щодо їх порядку, і пропонується інший канонічний представник - лексикографічна канонічна форма. Потім обговорюється феномен неподільності у випадку комбінованого сліду та його зв'язки з лексикографічною канонічною формою. У наступних розділах пропонується алгебраїчне представлення комбінованих слідів на основі проекцій на послідовні субалфавіти, і дається недетермінована процедура, яка дозволяє реконструювати послідовність кроків оригінального комбінованого сліду. Також

надаються дві стратегії визначення такої реконструкції, кожна з яких веде до належної канонічної форми комбінованого сліду. Зауважимо, що в даній роботі ми пропонуємо детермінований алгоритм, який реконструює звичайні сліди. . У заключному розділі описуються деякі природні застосування розроблених алгебраїчних властивостей та окреслюються напрями подальших досліджень. (8)

3. Основні теоретичні відомості

Напівгрупою називають множину, на якій задано бінарну асоціативну операцію. Моноїд - це напівгрупа з нейтральним елементом. Вільний частково комутативний (слідовий) моноїд – це моноїд, в якому деяким твірним дозволено комутувати, а деяким – ні. Точне визначення визначення слідового моноїда полягає в наступному. Нехай Σ – скінченний алфавіт. $\Sigma*$ −вільний моноїд, породжений множиною Σ, його елементи – слова з операцією конкатенації. Нехай $I \subseteq \Sigma \times \Sigma$- симетричне та антирефлексивне бінарне відношення на алфавіті Σ.(його називають відношенням комутативності або незалежності). Слідовий моноїд $M(\Sigma, I)$ є фактормоноїдом $\Sigma*$ за конгруенцією, породженою парами $ab = ba, (a, b) \in I$. Елементи слідового моноїду називають слідами, кожен слід є сукупністю слів з $\Sigma*$, таких що кожне з них може бути отримано з іншого перестановками сусідніх незалежних літер. Кожному моноїду $M(\Sigma, I)$ можна поставити у відповідність неорієнтований граф без петель, де вершини – літери з Σ, а ребра проведено між парами комутуючих вершин з I. Такий граф називається графом незалежності моноїда $M(\Sigma, I)$. Також розглядають відношення залежності $D = \Sigma \times \Sigma \setminus I$, що є доповненням I , і відповідний граф залежності.

Розглядаючи входження різних букв до слів, що задають сліди, зручно використовувати позначення $alph(w)$ для множини букв, що входять до слова (сліда) w, а також $|w|_a$ для кількості букв a у слові w.

Для роботи з вільними частково комутативними моноїдами корисними інструментами є лема проектування та лема Леві.

Нехай $A \subseteq \Sigma$ – підмножина, а $I_A = (A \times A) \cap I$ – відношення комутативності, яке є звуження I на A. Можемо визначити канонічний гомоморфізм $\pi(A) : M(\Sigma, I) \to M(A, I_A)$ стиранням зі сліда всіх букв, які не належать A. Так для $a \in \Sigma$ маємо:

$\pi_A(a) =$ a, якщо $a \in A$ і $\pi_A(a) = 1$, якщо $a \notin A$.

Особливим є випадок $A = \{a, b\}$, таке що $(a, b) \in D$. Лема про проекції стверджує, що кожен слід можна представити у вигляді кортежу слів-проекцій, а

саме.

Твердження 3.1 (лема про проекції) Нехай u та v– два сліди з $M(\Sigma, I)$. Тоді $u = v$ тоді і тільки тоді, коли

$\pi_{a,b}(u) = \pi_{a,b}(v)\ \forall(a, b) \in D$.

Одним з важливих тверджень в теорії слідів є лема Леві. Позначення $alph(x) = \{ a \in \Sigma | |x_a| \neq 0 \}$ використовується для множини літер з Σ, які є у слові x.

Твердження 3.2 (лема Леві): Нехай t, u, v, w – сліди з $M(\Sigma, I)$. Тоді наступні умови є еквівалентними:

1) $tu = vw$;

2) існують такі $p, q, r, s \in M(\Sigma, I)$, що $t = pr, u = sq, v = ps, w = rq$, а також $alph(r) \times alph(s) \subseteq I$.

3.1.Нормальні форми

Будемо вважати, що алфавіт Σ цілком впорядкований, і ми розглядаємо відповідний лексикографічний порядок $<$ на Σ^*. Нехай X – множина слів, унікальний мінімальний елемент X відносно лексикографічного порядку (якщо він існує) визначається як $Min(X)$. Говорять, що слово x знаходиться в лексикографічній нормальній формі, якщо воно мінімальне у множини слів, які еквівалентні x.

$$x = Min([x])$$

Так як лексикографічний порядок – це повний порядок і [x] скінченне, кожен слід має мінімального унікального представника. (3)

Теорема 4.1. Слово x є лексикографічною нормальною формою сліда, тоді і тільки тоді, коли для всіх факторизацій $x = ybuaz$, де $x, y, z \in \Sigma*, (a, b) \in I$, $a < b$ і існує буква з u, що не комутує з a.

Слово x з Σ^* знаходиться в нормальній формі Фоата, якщо це порожнє слово, або якщо існує ціле число $n > 0$ і непорожні слова $x_i (1 \leq i \leq n)$ такі, що:

1)$x = x_1 \dots x_n$

2)для кожного i слово x_i-це добуток попарно незалежних літер і x_i-

мінімальне відносно лексикографічного порядку

3)для кожного $1 \leq i \leq n$ і для кожної літери a з x_{i+1} існує літера b з x_i така, що $(a, b) \in D$.

Теорема 4.2. Кожен слід має унікальну нормальну форму Фоата. (3)

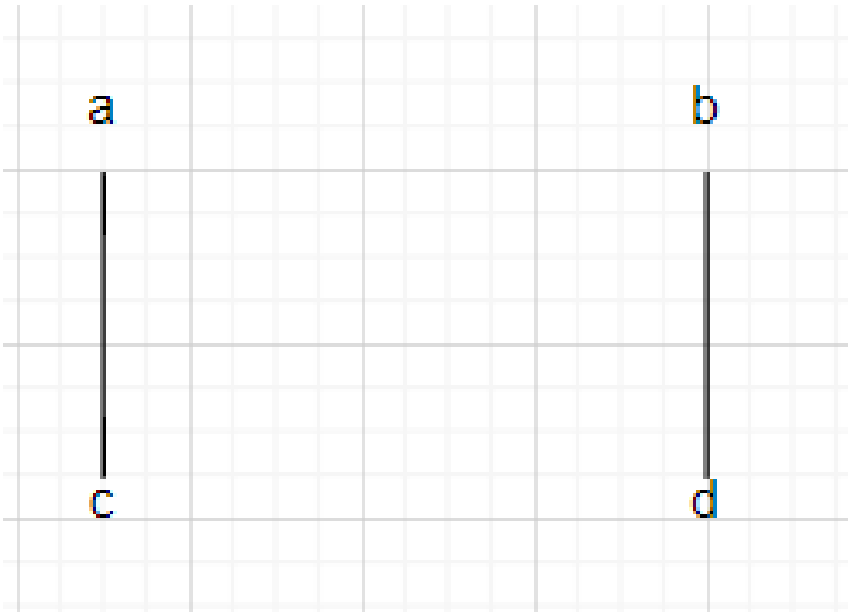
3.2.Простий алгоритм, щоб обчислювати нормальні форми

Опишемо простий метод, який дозволяє обчислити нормальні форми. Нехай $M(\Sigma, I)$ – вільний частково комутативний моноїд, ми використовуємо стек для кожної літери алфавіту Σ. Нехай x– слово з $\Sigma*$, ми скануємо x справа наліво; коли обробляється літера a, літера a виштовхується зі стеку і ставиться маркер на всі стеки літер, які не комутують з a, крім a.

-Щоб отримати лексикографічну нормальну форму, достатньо взяти з літер, які знаходяться на вершині стеку, що літера a є мінімальною у відношенні даного лексикографічного порядку. Ми виштовхуємо маркер на кожен стек, який відповідає літері, що не комутує з a, крім a. Ми повторюємо цей цикл доти, доки всі стеки не стануть порожніми.

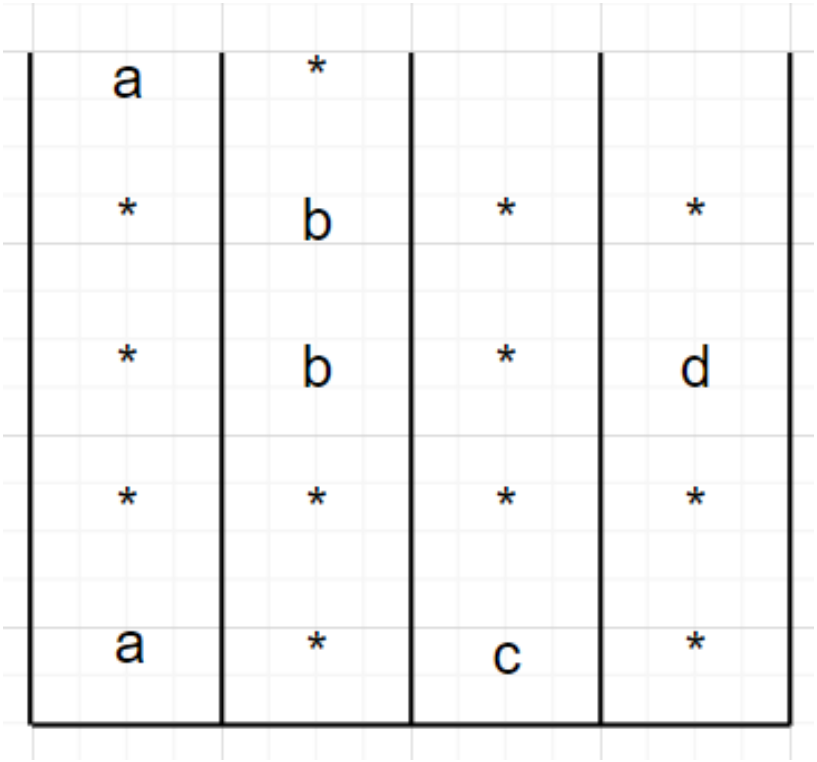
-Щоб отримати нормальну форму Фоата, ми беремо всередині циклу множину з літер, які знаходяться на вершині стеків; розташування літер в лексикографічному порядку дає крок. Цикл повторюється доти, доки всі стеки не спорожніють. (3)

Приклад:



Мал.1. (Σ,I)

Для прикладу з таким (Σ, I) і слова abdbca отримуємо наступні стеки, які приведено нижче Лексикографічна нормальна форма – abbdac, нормальна форма Фоата-(a)(bd)(b)(ac).



Мал.2. Стеки

4. Відновлення слідів за проекціями та розв'язання деяких класів рівнянь.

Згідно з лемою про проекції, два слова рівні в моноїді тоді і тільки тоді, коли для кожної пари букв(a, b) з відношення залежності D проекції цих слівпід дією $\pi_{a,b}$ співпадають. Це означає, що коли в нас є набір проекцій якогось слова, то слово можна однозначно по ньому відновити.

Отже, виникає низка природних питань:

- Чи кожен набір слів у вільних моноїдах $\{a, b\}^*$ $((a, b) \in D)$ задає деякий слід?
- Якщо ні, то якими є умови існування сліда з даним набором проекцій?
- Яким може бути алгоритм відновлення сліда за проекціями?

На ці питання ми будемо відповідати в даному розділі. А також застосуємо отримані алгоритми до розв’язку рівнянь певного вигляду.

4.1. Відновлення елемента моноїда за його проекціяхми

Домовимося розглядати $M = M(\Sigma, I)$ – слідовий моноїд, нехай D – його відношення залежності. Зауважимо, що, зокрема, $(a, a) \in D$ для кожного $a \in \Sigma$.

Спершу зазначимо, що не кожен набір слів у вільних моноїдах $\{a, b\}^*((a, b) \in D)$ задає деякий слід у моноїді $M(\Sigma, I)$, яке це показано у наступному прикладі.

Приклад. Розглянемо моноїд $M = M(\Sigma, I)$ з $\Sigma = \{a, b, c\}, I = \emptyset$, тобто вільний моноїд з трьома твірними. Для слів $w_{ab} = ab, w_{bc} = bc,\ w_{ac} = ca$ не існує елемента з M з відповідними проекціями, оскільки якби такий елемент існував, то жодна з букв a, b, c не могла б бути його першою буквою: a має стояти після c, c – пізніше b, а b в свою чергу пізніше a.

У той же час зазначимо декілька очевидних необхідних умов на "потенційні" проекції.

Твердження 4.1. Нехай задано набір слів $\{w_{ab}\}$ у вільних моноїдах $\{a, b\}^*((a, b) \in D)$. Якщо слова з цього набору є проекціями деякого сліду з $M(\Sigma, I)$, то

1) для кожної пари w_{ab}, w_{ac} виконано $|w_{ab}|_a$=$|w_{ac}|_a = |w_{aa}|_a$;

2) для кожної трійки a, b, c попарно залежних букв виконано наступне: якщо w_{ab} починається з букви a, а w_{bc} починається з букви b, то w_{ac} починається з букви a.

Спостереження з пункту 2) можна узагальнити в такий спосіб:

Нехай задано набір слів $\{w_{ab}\}$ у вільних моноїдах $\{a, b\}^*((a, b) \in D)$. Розглянемо відношення *передування* $\prec$ на множині Σ, визначене таким чином, що для різних a, b виконується $a \prec b$, якщо $(a, b) \in D$ і слово w_{ab} починається з a (тобто a стоїть в цьому слові раніше за з b). Також вважатимемо, що $a \prec a$ для всіх a.

Твердження 4.2. Нехай слова з набору $\{w_{ab}\}$ є проекціями деякого сліду з $M(\Sigma, I)$. Тоді віднощення $\prec$ має бути відношенням часткового порядку.

Ми використаємо Твердження 4.2. для побудови алгоритму відновлення сліду за його проекціями. Спочатку розглянемо частковий випадок, а саме відновлення слова за його проекціями у вільному моноїді. Ідея алгоритму полягає у послідовному відновленні букв вихідного слова, починаючи з першої.

Вхід: проекції слова на всі пари літер, моноїд: список літер та список комутуючих пар літер.

Вихід: або слово або повідомлення про те, що такого слова немає

Кроки алгоритму:

```
while projections:
//створення лінійного порядку
    for projection in projections:
        for letter in projection:
            if letter != projection[0]:
                hasse.append((projection[0],letter))
//пошук мінімального елемента
        for el in (x[0] for x in hasse):
            if el not in (x[1] for x in hasse):
                letter=el
```

```
        word+=letter
    //видалення цього елемента з черг проекцій
        for projection in projections:
         if  letter == projection[0]:
           projections.remove(projection)
           if len(projection)!=1:
              projections.append(projection[1:])
    return word
```

Алгоритм працює коректно, якщо на кожному кроці поточне відношення передування є лінійним порядком. У такому разі ми однозначно визначаємо поточну букву сліда як (єдиний) мінімальний елемент множини букв.

Складність алгоритму залежить лінійно від кількості проекцій.

Розглянемо роботу алгоритму на прикладі:

Вхід:

M – вільний моноїд, $\{a,b,c\}^*.\pi_{a,a}=aa,\quad \pi_{b,b}=b,\pi_{c,c}=cc,\pi_{a,b}=$ aab, $\pi_{a,c}=$ caac, $\pi_{b,c}=$ cbc.

1буква: a$\prec b, c \prec a, c \prec b$ – відношення лінійного порядку. c -мінімальний елемент діаграми Хассе. Таким чином, викреслюємо букву c з усіх проекцій, де вона стоїть першою. $\pi_{a,a}=aa$, $\pi_{b,b}=b,\pi_{c,c}=c,\pi_{a,b}=$ aab, $\pi_{a,c}=$ aac, $\pi_{b,c}=$ bc.

Слово:c

2буква: a≺b, a≺c,b≺c. Мінімальний елемент –a. Викреслюємо a.

$\pi_{a,a}=a$, $\pi_{b,b}=b,\pi_{c,c}=c,\pi_{a,b}=$ ab, $\pi_{a,c}=$ ac, $\pi_{b,c}=$ bc.

Слово:ca

3буква: a≺b, a≺c,b≺c. Мінімальний елемент – a. Викреслюємо a.

$\pi_{a,a}=$, $\pi_{b,b}=b,\pi_{c,c}=c,\pi_{a,b}=$ b, $\pi_{a,c}=$ c, $\pi_{b,c}=$ bc.

Слово:caa

4 буква: b≺c. Мінімальний елемент –b. Викреслюємо b.

$\pi_{a,a}=$, $\pi_{b,b}=$, $\pi_{c,c}=c,\pi_{a,b}=$, $\pi_{a,c}=$ c, $\pi_{b,c}=$ c.

Слово:caab

5 буква: c

x = caabc.

Вихід: caabc

Тепер перейдемо до загального випадку. На відміну від вільного моноїда, відношення $\prec$, що виникає на кожному кроці, не буде лінійним порядком (лише частковим), адже в цьому випадку для незалежних букв a, b не існує проекції w_{ab}, за якою можна було б порівняти a, b. З цього випливає, що замість відновлення сліда за буквами, варто відновлювати його за кроками нормальної форми Фоата, кожен з яких містить попарно незалежні букви.

```
        while projections:
//створення часткового порядку
        for projection in projections:
          for letter in projection:
             if letter != projection[0]:
                 hasse.append((projection[0],letter))
//пошук мінімальних елементів
        for el in (x[0] for x in hasse):
          if el not in (x[1] for x in hasse) and el not in letter:
             letter+=el
             word+=el
//якщо немає часткового порядку
        if len(hasse)==0:
           for el in projections:
              if el not in letter:
                 letter+=el
                 word+=el
//видалення літер з черг проекцій
        for lett in letter:
         for projection in projections:
            if  lett == projection[0]:
```

```
                projections.remove(projection)
                if len(projection)!=1:
                  projections.append(projection[1:])
        return word
```

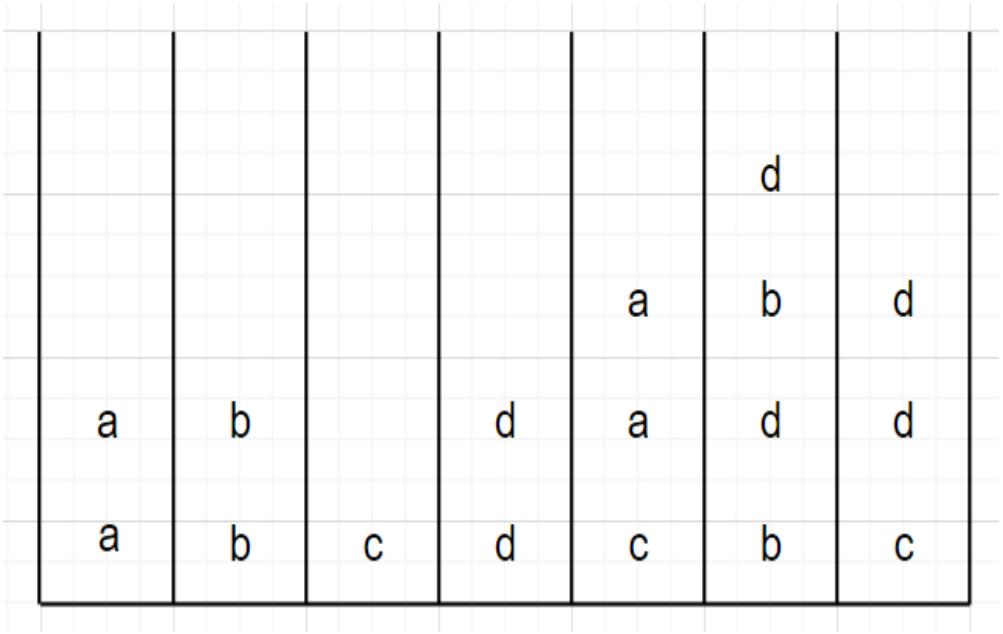
Доведемо коректність роботи алгоритму. Зазначимо, що a, b є порівнюваними у відношенні передування $\prec$ тоді й лише тоді, коли $(a, b) \in D$. Це означає, по-перше, що мінімальні елементи відношення $\prec$, що виникають на кожному кроці роботи алгоритму, є попарно незалежними елементами. Якби це було б не так, то для пари $(a, b) \in D$ знайшлося б слово-проекція w_{ab}, за яким можна було б порівняти a, b, а отже, одне з них не могло б бути мінімальним. По-друге, це означає, що коли a є одним з мінімільних елементів $\prec$ на поточному k-ому кроці (належить множині MIN_k), то обов'язково знайдеться деяке b з мінімальних елементів попереднього кроку (множини MIN_{k-1}), таке що $(a, b) \in D$. Інакше a було б непорівнюваним з кожним елементом MIN_{k-1}, а тому a мало б бути мінімальним елементом і належати MIN_{k-1}. Але в цьому випадку можна покласти $b = a$.

Розглянемо роботу алгоритма на прикладі.

$$I = \{(a, b), (a, d), (b, c)\}$$

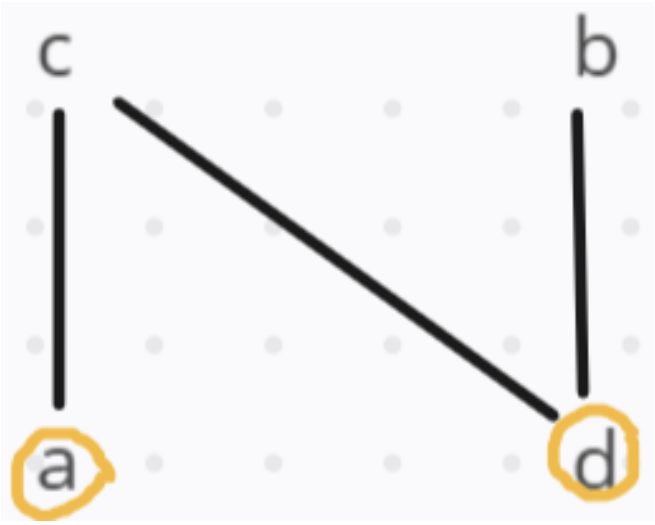
Мал.3. I

$$\pi_{a,a} = aa\ , \pi_{b,b} = bb, \pi_{c,c} = c, \pi_{d,d} = dd, \pi_{a,c} = aac, \pi_{b,d} = dbdb, \pi_{c,d} = ddc.$$



Мал.4. Стек напочатку

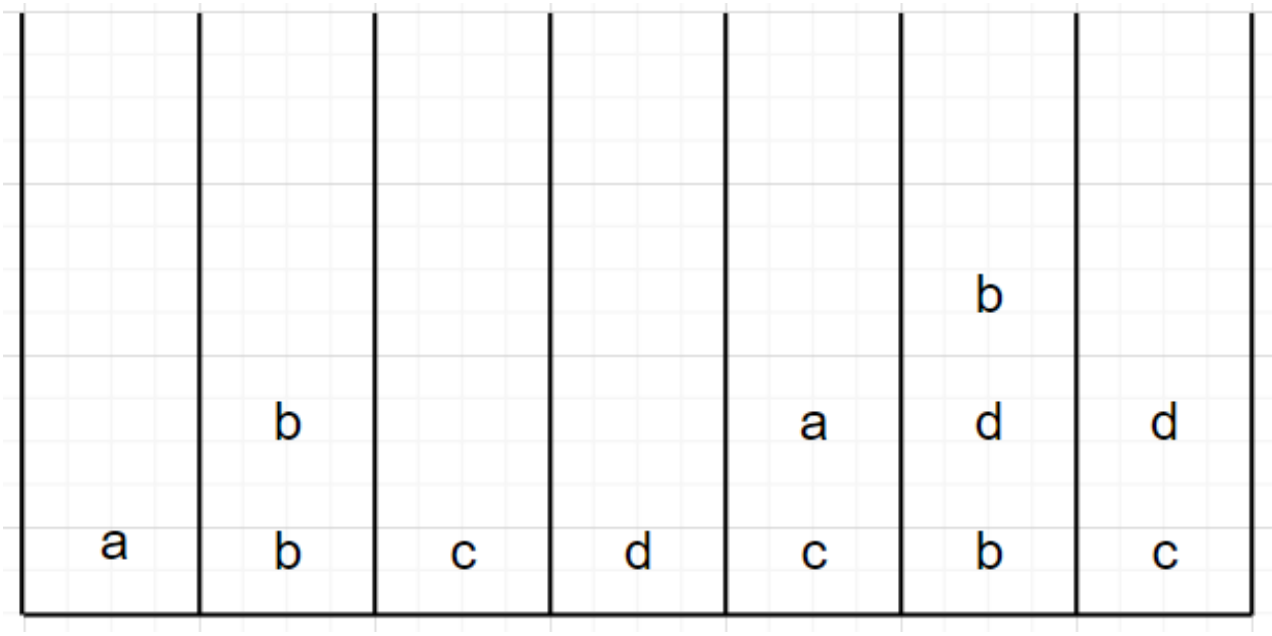
1 крок:a≺c, d≺b,d≺c, порядок не є лінейним. a і d -мінімальні елементи. Вони непорівнювані, так як немає проекцій на ці дві букви, а це означає, що вони комутують.



Мал.5. Діаграма Хассе - крок 1

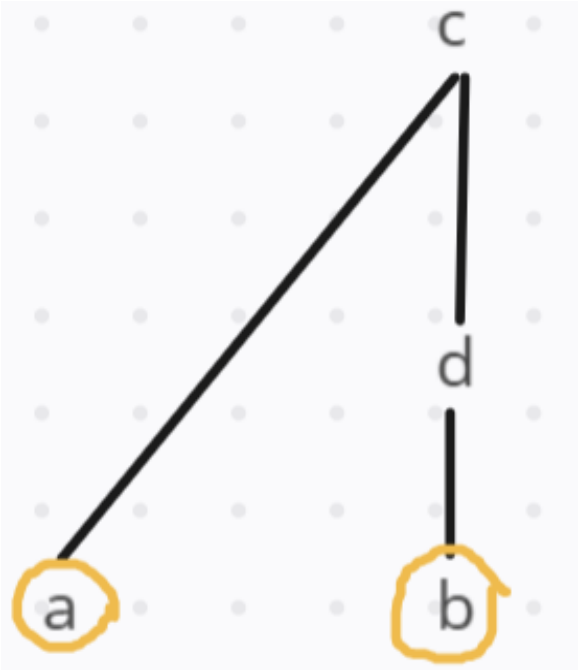
Слово: ad

$\pi_{a,a} = \text{a}, \pi_{b,b} = bb, \pi_{c,c} = c, \pi_{d,d} = d, \pi_{a,c} = \text{ac}, \pi_{b,d} = \text{bdb}, \pi_{c,d} = \text{dc}.$



Мал.6. Стек після кроку 1

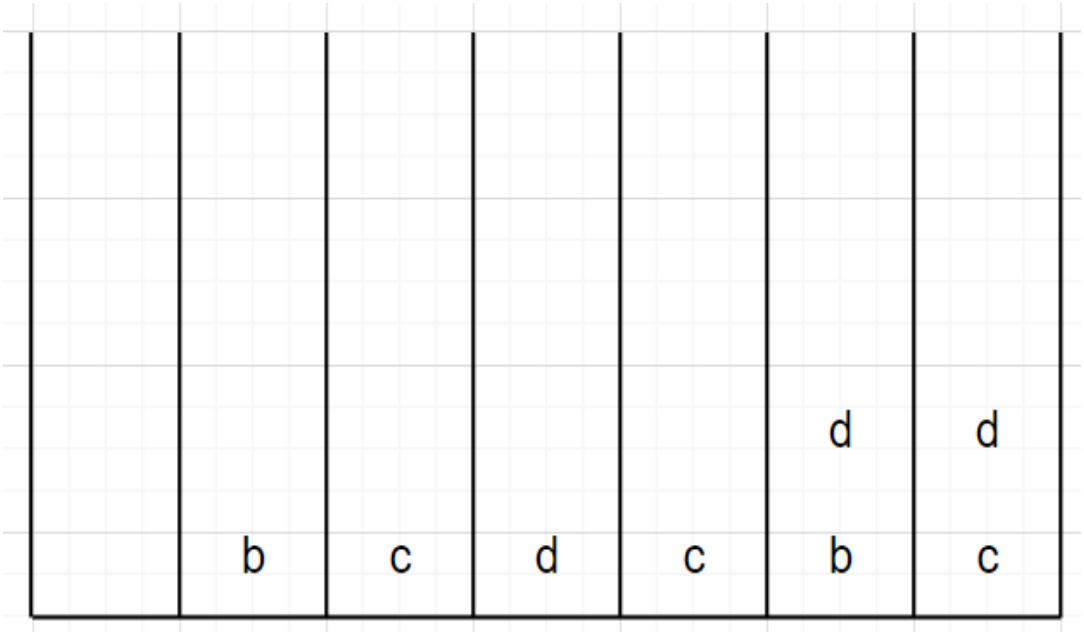
2 крок: $a \prec c, b \prec d, d \prec c$. a, b– мінімальні елементи.



Мал.7. Діаграма Хассе - крок 2

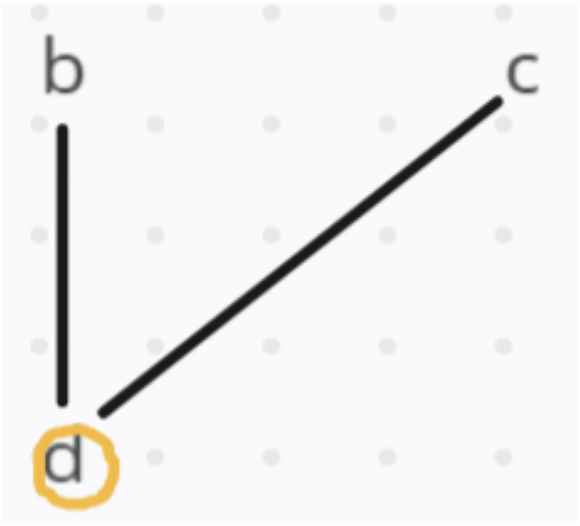
Слово:(ad)(ab)

$\pi_{a,a}$=, $\pi_{b,b} = b, \pi_{c,c} = c, \pi_{d,d} = d, \pi_{a,c} = c, \pi_{b,d} = db, \pi_{c,d} = dc.$



Мал.8. Стек після кроку 2

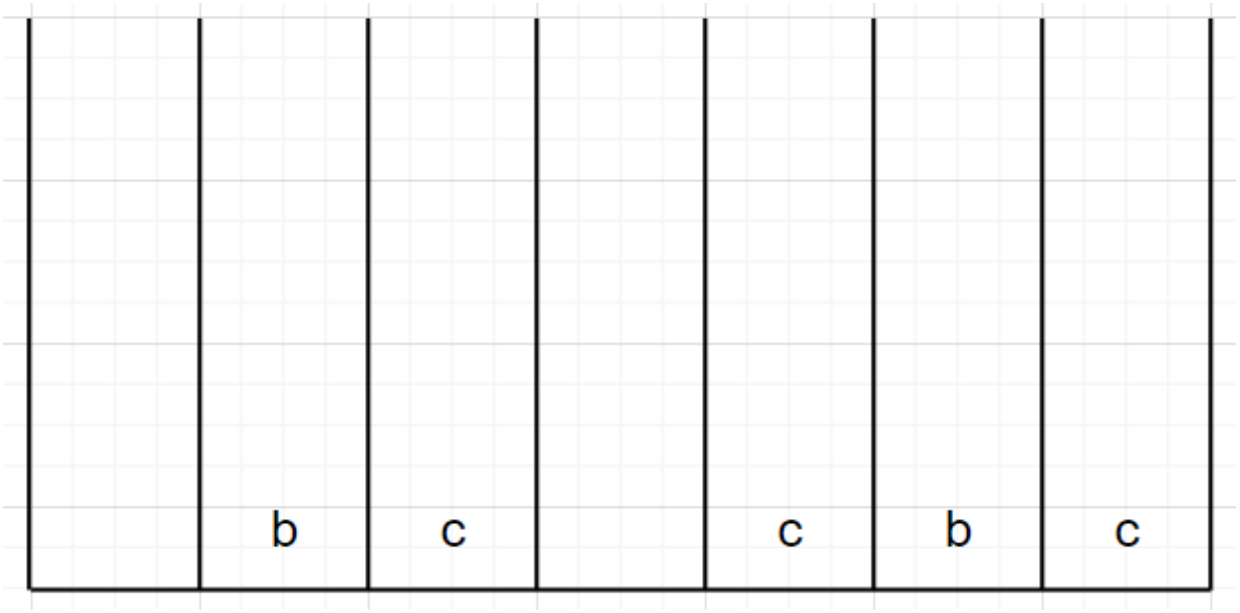
3 крок: $d \prec b, d \prec c$, d– мінімальний елемент.



Мал.9. Діаграма Хассе - крок 3

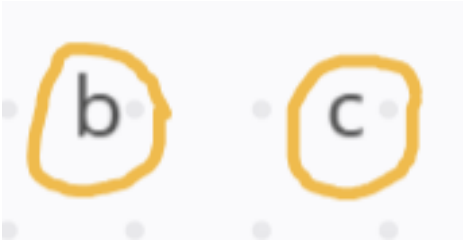
Слово: (ad)(ab)(d)

$\pi_{a,a}$=, $\pi_{b,b} = b, \pi_{c,c} = c, \pi_{d,d} =, \pi_{a,c} = c, \pi_{b,d} = b, \pi_{c,d} = c.$



Мал.10. Стек після кроку 3

4 крок: Залишилися тільки b і c, які не можна порівняти між собою.



Мал.11. Діаграма Хассе - крок 4

Слово: (ad)(ab)(d)(bc).

4.2. Розв'язання рівнянь вигляду $x^n = u$

Зауважимо, що у вільних моноїдах рівняння вигляду $x^n = u$ розв'язуються тривіальним способом. А саме, якщо $u = a_1 a_2 \dots a_l$, де $a_j \in \Sigma$, то рівняння має розв'язок в тому і лише тому випадку, коли $l = nk$ для деякого натурального k і $a_1 \dots a_k = a_{k+1} \dots a_{2k} = \dots = a_{(n-1)k} \dots a_{nk}$, тобто слово u розбивається на n однакових підслів.

Далі перейдемо до випадку слідового моноїда. Спочатку розглянемо частковий випадок, а саме квадратне рівняння вигляду $x^2 = u$, яке будемо розв'язувати відносно x у слідовому моноїді $M(\Sigma, I)$. Необхідною умовою існування розв'язку такого рівняння є

Твердження 4.2.1 . Якщо існує слід x, що задовольняє рівняння $x^2 = u$, то

$\forall a \in \Sigma \ |u|_a \equiv 0 \, mod \, 2.$

Також зазначимо, що оскільки проекції $\pi_{a,b}$ є гомоморфізмами, то для кожної пари $(a, b) \in$ D має виконуватися $\pi_{a,b}(u) = \pi_{a,b}(x^2) = \pi_{a,b}(x)^2$. Зважаючи на те, що $\pi_{a,b}(u)$ – слово у вільному моноїді, маємо наступне:

Твердження 4.2.2 . Якщо існує слід x, що задовольняє рівняння $x^2 = u$, то для кожної пари $(a, b) \in$ D слово $\pi_{a,b}(u)$ можна подати у вигляді добутку двох однакових слів з вільного моноїду $\{a, b\}^*$.

Природним чином ці необхідні умови узагальнюються на довільне натуральне $n \geq 2$, а саме має місце

Твердження 4.2.3 . Якщо існує слід x, що задовольняє рівняння $x^n = u$, то

1) $\forall a \in \Sigma \ |u|_a \equiv 0 \, mod \, n$;

2) для кожної пари $(a, b) \in$ D існує слово $w_{ab} \in \{a, b\}^*$, таке що $\pi_{a,b}(u) = w_{ab}^n$

Останнє твердження дає змогу сформулювати алгоритм розв'язання рівняння $x^n = u$.

1. Знайти проекції $\pi_{a,b}(u)$ для всіх $\pi_{a,b}(u)$.

2. Для кожної проекції $\pi_{a,b}(u)$ побудувати $w_{ab} \in \{a, b\}^*$, таке що $\pi_{a,b}(u) = w_{ab}^n$, використовуючи зауваження про розв'язання степеневих рівнянь у вільному моноїді.

3. За отриманими $w_{ab} \in \{a, b\}^*$ відновити x за допомогою алгоритма відновлення за проекціями.

Приклад:

$\pi_{a,b}(u) = ababab, \pi_{b,c}(u) = bcbcbc, \pi_{a,c}(u) = acacac, \pi_{a,a} = aaa, \pi_{b,b} = bbb, \pi_{c,c} = ccc$

$w_{ab} = ab, w_{bc} = bc, w_{ac} = ac, w_{aa} = a, w_{bb} = b, w_{cc} = c$

Слово: abcabcabc

4.3. Розв'язання рівнянь вигляду $xu = vx$

Цей розділ ми завершимо розглядом рівнянь ще одного типу, а саме $xu = vx$. Нагадаємо, що елементи u, v моноїда називають спряженими, якщо існує

такий елемент моноїда z, що $uz = zv$, тобто розв'язання рівняння $xu = vx$ рівносильно встановленню спряженості елементів u, v . Поняття спряженості є дослідженим в літературі, а саме існує необхідна і достатня умова спряженості (а отже, й існування розв'язку в рівняння $xu = vx$):

Твердження 4.3.1 [(3)

Нехай $u, v, z \in \mathrm{M}(\Sigma, \mathrm{I})$ – сліди. Тоді ми маємо $uz = zv$ тоді і тільки тоді, коли є сліди $z_1, z_2, u_1, \dots, u_k, k \geq 0$, які задовольняють наступним умовам:

1) $u = u_1 \dots u_k, v = u_k \dots u_1$

2) $u_i I u_j$ для $1 \leq i < j - 1 \leq k - 1$,

3) $z_1 = (u_1 \dots u_{k-1}) \dots (u_1 u_2) u_1$,

4)$z = z_1 z_2$ з $u I z_2$

Для алгоритмічного встановлення того, чи існує розв'язок у рівняння $xu = vx$ корисним може виявитися наступне

Твердження 4.3.2 [(3)]. Нехай граф відношення залежності D слідового моноїда $M(\Sigma, I)$ має діаметр d. Тоді сліди $u, v \in \mathrm{M}(\Sigma, \mathrm{I})$ спряжені тоді і тільки тоді, коли вони можуть бути трансформовані один в одного не більше, ніж за d транспозицій.

Приклад: $u = abcd, v = dcba$

Алгоритми відновлення слова по його проєкціях для випадку вільного та частково комутативного моноїдів було реалізовано на мові Python. Деталі ми обговоримо в наступному розділі.

5. Алгоритмічна реалізація відновлення слідів за проекціями та розв'язання рівнянь.

Реалізацію наведеного нижче описання архітектури можна знайти за адресою (9) .

Вікторією Ковальовою [(10)] було створено декілька класів, які представляють сутності предметної області:TraceMonoid та Word.

Клас Word описує слово. Для нього реалізовані наступні методи:

-знаходження проекції слова для двох заданих літер;

-метод reversed;

-метод calculate_prefix_conuter знаходить кількість заданих літер на префіксі слова;

-метод create_indices_mapping для знаходження відображення слова до літер іншого слова;

-метод delete_lex_min_subsequence для видалення лексикографічно мінімальної підпослідовності;

Клас TraceMonoid описує слід, заданий алфавітом та відношенням залежності. Для нього реалізовані наступні методи:

-words_are_equivalent – перевіряє еквівалентність двох слів;

-visulize_dependency – робить візуалізацію моноїда у формі графа залежності.

Для реалізації задач даної роботи було створено клас ElementOfMonoid з методами:

-reconstruction – для відновлення слова за його проекціями у вільному моноїді;

-reconstructionPartiallyComutative – для відновлення нормальної форми Фоата слова у частково комутативному моноїді.

Пакет розроблено за допомогою мови програмування Python3.7.

У файлі test.py наведено декілька прикладів використання цих двох функцій.

Наприклад,

```
element = ElementOfMonoid(["a","b","c"],{"a":"a","b":"b","c":"c"})
list=["aa","b","cc","aab","caac","cbc"]
print(element.reconstruction(list))
```

Результат:

```
caabc

Process finished with exit code 0
```

Мал.12. Приклад для вільного моноїда

```
el=ElementOfMonoid(["a","b","c","d"],{"c":"d","b":"d","a":"a","b":"b","c":"c","d":"d"})
list=["aa","bb","cc","dd","acac","adad","abab","bcbc"]
print(el.reconstructionPartiallyCommutative(list))
```

Результат:

```
abdcabdc

Process finished with exit code 0
```

Мал.13. Приклад для частково комутативного моноїда

6.Висновки

У роботі розглянуто основні поняття з теорії слідових моноїдів і зроблено огляд літератури з тематики розв’язання рівнянь у цих моноїдах. А також отримано наступні результати:

- Знайдено деякі необхідні умови існування сліду з заданими набором проекцій.
- Запропоновано алгоритм відновлення сліду за його проекціями для випадку вільних та вільних та частково комутативних моноїдів.
- Реалізовано алгоритм відновлення сліду за допомогою мови Python, з реалізацією можна ознайомитися за репозиторієм (9)
- Запропоновано техніку розв’язання рівнянь вигляду $x^n = u$ за допомогою алгоритму відновлення сліду за проекціями.

Список літератури

1. **Foata, Dominique та Cartier, Pierre.** R'earrangements de suites. *Probl'emes combinatoires de commutation et r'earrangements*. 1969 p., cc. 39-53.
2. **Mazurkiewicz, Antoni.** Concurrent program schemes and their interpretations. *DAIMI Report Series*. 1977 p.
3. **Diekert, Volker та Yves, M'etivier.** Partial commutation and traces. *Handbook of Formal Languages. Beyond Words*. 1997 p.
4. **Diekert, Volker.** Combinatorics on Traces. *Springer Science & Business Media*. 1990 p.
5. **Diekert, Volker та Muscholl, Anca.** Solvability of equations in free partially commutative groups is decidable. *Automata, Languages and Programming*. 2001 p., cc. 543-554.
6. **Diekert, Volker та Kufleitner, Manfred.** A Remark about Quadratic Trace Equations. *Springer*. 2002 p., cc. 59-66.
7. **Robson, John Michael та Volker, Diekert.** On quadratic word equations. *Springer*. 2002 p., cc. 217-226.
8. **Mikulski, Lukasz.** Algebraic structure of combined traces. *Logical Methods in Computer Science*. 2013 p., cc. 1-26.
9. **Mysyk, Oleksandra.** [Онлайновий] 1 6 2024 p. https://github.com/AlexandraMysyk/trace_monoids.
10. **Kovalova, Viktoria.** Trace_monoids. *GitHub*. [Онлайновий] 31 5 2021 p. https://github.com/ViktoriiaKovalova/trace_monoids.
11. *40th Annual Symposium on Foundations of Computer Science*. **Plandowski, W.** 1999. Satisfiability of word equations with constants is in PSPACE.
12. **Mikulski, Lukasz.** Projection representation of Mazurkiewicz traces. *Fundamenta Informaticae*. 2008 p.
13. **Моноїд. *Wikipedia*. [Онлайновий] 24 1 2022 p. https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%BD%D0%BE%D1%97%D0%B4.**
14. Trace monoid. *Wikipedia*. [Онлайновий] 24 10 2022 p. https://en.wikipedia.org/wiki/Trace_monoid.
15. Hilbert's tenth problem. *Wikipedia*. [Онлайновий] 17 1 2024 p. https://en.wikipedia.org/wiki/Hilbert%27s_tenth_problem.
16. Duboc, Christine. On Some Equations in Free Partially Commutative Monoids. 1986 p.
17. Asveld, Peter R.J. Controlled iteration grammars and full hyper-AFL's. *Information and control*. 1977 p., cc. 177-269.
18. *Meeting on String Constraints and Applications*. Diekert, Volker. 2019.
19. *43rd International Colloquium on Automata, Languages and Programming*. Diekert, Volker, Jez, Artur та Kufleitner, Manfred. 2016. Solutions of Word Equations Over Partially Commutative Structures.
20. Diekert, Volker, Gutierrez, Claudio та Hagenah, Christian. The existential theory of equations with rational constraints in free groups is PSPACE-complete. *Springer*. 2001 p.