КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра теоретичної кібернетики

Кваліфікаційна робота

на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки

на тему:

РОЗРОБКА ІНТЕРПРЕТАТОРА МАШИН ТЮРІНҐА

(DEVELOPMENT OF THE TURING MACHINE INTERPRETER)

Виконав студент 4-го курсу групи ТК-41
Владислав ГОНЧАР

(підпис)

Науковий керівник:
Доцент, кандидат фіз.-мат. наук
Андрій СТАВРОВСЬКИЙ

(підпис)

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

Студент

(підпис)

Розглянуто й допущено до захисту
на засіданні кафедри теоретичної кібернетики
«01» червня 2022 р., протокол № 11
Завідувач кафедри
доктор фіз.-мат. наук, професор
Юрій КРАК

(підпис)

Київ – 2022

РЕФЕРАТ

Обсяг роботи 38 сторінок, 14 ілюстрацій, 2 таблиці, 11 джерел посилань.

ВЕБЗАСТОСУНОК, ВІЗУАЛІЗАЦІЯ МАШИН ТЮРІНҐА, ІНТЕРПРЕТАТОР МАШИН ТЮРІНҐА, МАШИНИ ТЮРІНҐА, НАЧАЛЬНА ПРОГРАМА, ПРОБЛЕМА ЗУПИНКИ, ТЕОРІЯ АЛГОРИТМІВ

Об’єктом роботи є процес розробки програмного продукту – інтерпретатора машини Тюрінґа, який буде використовуватись у навчальних закладах вищої освіти.

Метою розробки є написання якісної програми, що буде здатна інтерпретовувати машини Тюрінґа, буде володіти якостями програми, що може бути використана в освітньому процесі.

Методи розробки: використання програмних засобів, за допомогою яких можна розробити програмний продукт; вивчення об’єкту розробки та інформації про машини Тюрінґа задля отримання якісного результату роботи; огляд сучасних методів навчання, аналіз підходів до навчального процесу і впровадження найкращих з них у розробленому продукті. Інструменти розробки: текстовий редактор Visual Studio Code, програмне середовище Node.js.

Результати роботи: було розроблено програмний продукт, який здатний інтерпретувати машини Тюрінґа, може бути використаний у навчальних закладах задля покращення процесу навчання у проходженні та засвоєнні дотичних до машин Тюрінґа тем.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ІМТ – імовірнісна машина Тюрінґа
МТ – машина Тюрінґа
НДМТ – недетермінована машина Тюрінґа
ПЗ – програмне забезпечення
ПК – персональний комп’ютер
УМТ – універсальна машина Тюрінґа

ВСТУП

Оцінка сучасного стану об’єкта розробки. Останнім часом освітній процес в Україні й у світі зазнав значних змін. З приходом пандемії коронавірусної хвороби 2019, усе більше приділяється уваги дистанційному навчанню на противагу традиційним методам отримання освіти, коли учні (студенти, здобувачі освіти) навчались безпосередньо в будинках і спорудах закладів освіти. Насамперед, звісно, причиною переходу на дистанційну освіту є міркування безпеки, але вже зараз можна сказати, що навіть коли пандемія остаточно піде спад, то багато елементів освіти і способів навчання, що тимчасово прийшли на заміну консервативним підходам до здобуття освіти, залишаться на тривалий і довгий час, аж поки не з’явиться альтернатива і до таких підходів навчання.

Виникає питання, а у чому ж тоді перевага нових методів навчання і як це може стосуватися розробки інтерпретатора машин Тюрінґа? Відповідь на таке питання може бути тільки комплексною, тому спочатку треба сказати, що з машинами Тюрінґа та їхнім застосуванням найчастіше студенти знайомляться у таких навчальних дисциплінах, як дискретна математика, математична логіка або теорія алгоритмів. У цих дисциплінах тема машин Тюрінґа часто-густо йде поруч із машинами Поста та машинами з натуральнозначними регістрами (обидві машини є Тюрінґ-повними), що по своїй суті є абстрактними моделями обчислення, які є дуже подібними до комп’ютерів, але з деякими відмінностями – такими, як наприклад, послідовний, а не довільний доступ до пам’яті. Сухе і суто теоретичне вивчення матеріалу, що може мати застосування в інформаційній сфері (інформаційно-навчальне програмне забезпечення), не буде давати тих результатів, що ставлять перед собою викладачі й розробники програм навчальних дисциплін, оскільки буде втрачатися одна з основних цілей навчання – пояснення мети вивчення тієї чи іншої теми та можливості застосування знань щодо предмету навчання. Сильною перевагою нових методів

навчання можна назвати широке використання навчальних комп’ютерних програм. Складно не погодитись, що використання програми, що, наприклад, інтерпретує якусь математичну модель (таку, як наприклад, машина Тюрінґа) полегшить сприйняття й пізнавання інформації здобувачем освіти.

Актуальність роботи та підстави для її виконання. Спроба покладатись лише на уяву при вивченні такої теми, як машини Тюрінґа, в інформаційну еру може визивати тільки подив. Візуалізація і інтерпретація роботи таких складних абстрактних обчислювальних машин є необхідністю у сьогоденні.

По-перше, написання програм для машин Тюрінґа є складною задачею, а написання правильних програм (що відповідають задуму того, хто її написав) є ще більш складною задачею. Інтерпретація роботи машини на папері може перетворитись на суцільне пекло, водночас використанням програми-інтерпретатора – це питання декількох секунд.

По-друге, використання візуалізації робить сприйняття і розуміння інформації швидшим, що дає можливість приділити більше уваги на деталі предмету вивчення.

Мета й завдання роботи. Метою кваліфікаційної роботи є розробка програмного забезпечення, що буде здатне інтерпретувати машини Тюрінґа. Задля успішного виконання роботи, треба:

- Проаналізувати сучасний стан засобів навчання теорії алгоритмів
- Дослідити різні моделі машин Тюрінґа
- Розробити програмний продукт
- Розробити графічний інтерфейс до програми
- Проаналізувати результат роботи

Об’єкт, методи й засоби розробки. Об’єктом розробки програмного забезпечення є інтерпретатор машин Тюрінґа. Програма буде розроблена для навчальних закладів, тому потребує якісної реалізації. Буде використаний найбільш підхожий сучасний метод розробки, щоби фінальний продукт міг бути модифікованим. Програма буде написана мовою JavaScript, у середовищі

Node.js.

Можливі сфери застосування. Розроблений продукт можна використовувати у сфері навчання у закладах вищої освіти. Окрім використання в закладах освіти, продукт можна використовувати для написання програм для машин Тюрінґа.

РОЗДІЛ 1. ПОНЯТТЯ МАШИНИ ТЮРІНҐА

1.1 Історія створення машини Тюрінґа

У 1928 році Давид Гільберт сформулював проблему розв’язності[1]: чи можна знайти алгоритм, який би як вхідні дані опис формальної мови та деяке математичне твердження цією мовою, і після скінченного числа кроків робив зупинку й давав відповідь «Істина» або «Хиба» відповідно до істинності заданого твердження, причому алгоритм повинен був давати правильну відповідь.

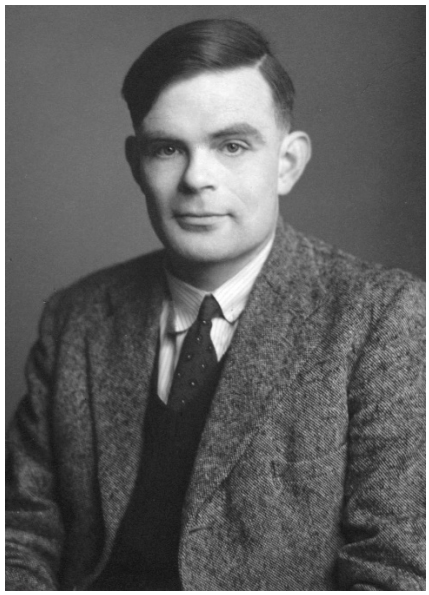

Рисунок 1.1 – Алан Тюрінґ

Під час навчання у Прінстонському університеті англійський математик Алан Тюрінґ у 1936 р. написав роботу «Про обчислювані числа із застосуванням до проблеми розв’язності», яка стала основою інформатики. У ній Тюрінґ довів, що такого алгоритму не існує, використовуючи створений ним математичний апарат – абстрактну машину[2][3], яка могла б розв’язати будь-яку проблему, яку можна було описати простими інструкціями, закодованими на нескінченній стрічці. Одна машина Тюрінґа, наприклад, могла обчислювати цілі квадратні корені, а інша могла порівнювати числа. Тюрінґ продемонстрував, що можливо побудувати єдину універсальну машину, яка могла б імітувати будь-яку машину Тюрінґа. Одна машина, яка вирішує будь-яку проблему, виконує будь-яке завдання, для якого можна написати програму. По-суті, Алан Тюрінґ, даючи відповідь на проблему розв’язності, винайшов комп’ютер.

1.2 Опис машини Тюрінґа

Машина Тюрінґа, або обчислювальна машина, як її назвав Тюрінґ, в оригінальному визначенні - це машина, яка містить скінченну множину конфігурацій $q_1, \dots, q_n$ (станів, які Тюрінґ називав m-конфігураціями). Машина обладнана нескінченною в обидва боки одновимірною стрічкою, поділеною на клітини. Клітина містить будь-який символ із алфавіту стрічки.

У момент роботи машина сканує вміст поточної клітини. Поведінка машини повністю визначена поточним станом (m-конфігурацією) і символом, що записаний у поточній клітині. Машина Тюрінґа здатна на такі 3 типи поведінки. Перший – це записати у комірку символ алфавіту S_i, перейти на одну комірку вліво, перейти до стану q_j, другий - записати у комірку символ алфавіту S_i, перейти на одну комірку вправо, перейти до стану q_j, і третій – зупинитись.

Кожну машину Тюрінґа можна задати скінченною множиною кортежів з п’яти елементів, а саме: $q_i S_j S_k M_n q_m$, де q_i – поточний стан, S_j – символ алфавіту стрічки, що був прочитаний, S_k – символ алфавіту, який буде записаний у поточну комірку, M_n – напрямок руху по стрічці, q_m – стан, у який перейде машина після руху по стрічці.

Наприклад, задамо машину Тюрінґа, що записує на стрічку послідовність $S_0 S_0 S_1 S_0 S_0 S_1 \dots$:

$$q_0 S_0 S_0 R q_1$$
$$q_0 S_1 S_0 R q_1$$
$$q_1 S_0 S_0 R q_2$$
$$q_1 S_1 S_0 R q_2$$
$$q_2 S_0 S_1 R q_0$$
$$q_2 S_1 S_1 R q_0$$

Насправді, можна прибрати інструкції, які прочитують (сканують) символ алфавіту S_1, бо вони є надлишковими. Вони були введені для більш наочного представлення таблиці переходів, а саме:

	S_0	S_1
q_0	S_0Rq_1	S_0Rq_1
q_1	S_0Rq_2	S_0Rq_2
q_2	S_1Rq_0	S_1Rq_0

Табл. 1.2.1

Велика кількість можливостей машини Тюрінґа полягає в тому, що якщо, наприклад, деякі алгоритми *A* та *B* імплементуються машинами Тюрінґа, то можна писати програми машин Тюрінґа, що являють собою композиції програм (алгоритмів). Нехай у нас є якісь алгоритми *A* і *B*. Тоді, якщо машина Тюрінґа може виконати алгоритм *A*, то залежно від результату виконання можна виконувати наступний алгоритм, наприклад, *B* і так далі – тобто алгоритми можна комбінувати.

Поєднання різних алгоритмів є алгоритмом. Тому їхня реалізація за допомогою машини Тюрінґа служить одним із засобів обґрунтування універсальності конструкції Тюрінґа.

Машина Тюрінґа є загальним прикладом центрального процесора (ЦП), який керує всіма маніпуляціями з даними, що здійснюються комп'ютером, при цьому машина Тюрінґа використовує послідовну пам'ять для зберігання даних. Точніше, це машина (автомат), яка здатна перераховувати деяку довільну підмножину рядків алфавіту; ці рядки є частиною рекурсивно перелічної множини.

Розглядаючи машину Тюрінґа як чорну скриньку, ми знаємо, що неможливо знати, чи врешті-решт вона перелічить будь-який конкретний рядок підмножини даної програми. Це пов’язано з тим, що проблема зупинки є нерозв’язною, що має серйозні наслідки, що полягають в обмеженнях у формальній теорії обчислень.

Машина Тюрінґа, яка здатна імітувати будь-яку іншу машину Тюрінґа, називається універсальною машиною Тюрінґа або просто універсальною машиною. Математичне визначення було введене Алонзо Черчем, чия робота з лямбда-численням переплiталася з теорією Тюрінґа у формальній теорії

обчислень, відомій як теза Черча-Тюрінґа. У дисертації стверджується, що машини Тюрінґа дійсно охоплюють неформальне поняття ефективних методів у логіці та математиці та дають точне визначення алгоритму або «механічної процедури». Вивчення їх абстрактних властивостей дає багато уявлень про інформатику та теорію складності обчислень.

1.3 Формальне визначення машини Тюрінґа

Формально машину Тюрінґа можна визначити як кортеж з чотирьох множин, двох особливих елементів з двох множин із попередніх чотирьох, та часткової функції переходів, а саме: Γ – непуста скінченна множина символів, що може бути записана на стрічку під час роботи машини Тюрінґа; $\sum \subseteq \Gamma \setminus \{b\}$ – скінченна множина символів, що можуть бути на стрічці до початку роботи машини Тюрінґа, які водночас не є пустими символами; $b \in \Gamma$ - елемент скінченної множини символів Γ, що позначає пустий символ; Q – непуста скінченна множина станів, у яких може перебувати машина Тюрінґа; $q_0 \in Q$ – елемент скінченної множини станів Q, що позначає початковий стан машини Тюрінґа; $F \subseteq Q$ – скінченна множина кінцевих станів, після переходу в які машина Тюрінґа зупиняється.

Функція $\delta: (Q \setminus F) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ – часткова функція, яку називають *функцією переходів*. Вона є частковою, бо не визначена на множині кінцевих станів F, бо перейшовши у стан із множини кінцевих станів машина Тюрінґа зупиняється. Функція переходів позначає, до якої комірки на стрічці та в який стан перейде машина Тюрінґа, прочитавши символ S, перебуваючи у стані $q \in Q \setminus F$.

Опишемо попередній приклад машини Тюрінґа формально. $\Gamma = \{S_0, S_1\}$, $b = S_0$, $\sum = \{\}$, $Q = \{q_0, q_1, q_2\}$, q_0 – початковий стан, $F = \{\}$, δ – задається таблицею 1.2.1

РОЗДІЛ 2. НЕДЕТЕРМІНОВАНА МАШИНА ТЮРІНҐА

2.1 Опис недетермінованої машини Тюрінґа

Недетермінована машина Тюрінґа — це теоретична модель обчислень, правила якої визначають більше ніж одну можливу дію в деяких заданих ситуаціях. Тобто наступний стан недетермінованої машини Тюрінґа не повністю визначається його станом та поточним символом, який він зчитав, на відміну від детермінованої машини Тюрінґа.

Недетермінована машина Тюрінґа іноді використовуються в мисленнєвих експериментах для вивчення можливостей і обмежень комп'ютерів. Однією із найбільш важливих проблем сучасності є проблема P проти NP, яка (серед інших еквівалентних формулювань) стосується питання про те, наскільки важко моделювати недетерміновані обчислення за допомогою детермінованого комп'ютера.

По суті, машину Тюрінґа уявляють як простий комп'ютер, який читає та записує символи по одному на нескінченній стрічці, суворо дотримуючись набору правил. Він визначає, яку дію він повинен виконати далі відповідно до свого внутрішнього стану та який символ він зараз бачить. Прикладом одного з правил машини Тюрінґа може бути такий: «Якщо ви перебуваєте в стані 2 і зчитали «A», змініть його на «B», перемістіть ліворуч і перейдіть до стану 3».

На відміну від детермінованої машини Тюрінґа, в недетермінованій машині Тюрінґа набір правил може передбачати більше однієї дії, яка має виконуватися для будь-якої даної ситуації. Наприклад, X на стрічці в стані 3 може дозволити недетермінованій машині Тюрінґа виконати такі кроки: «Напишіть Y, перейдіть вправо та перейдіть у стан 5» або «Напишіть X, перемістіться ліворуч і залишайтеся в стані 3».

Як недетермінована машина Тюрінґа «знає», які з цих дій йому слід зробити? Є два способи обробити таку ситуацію. Одна з них — це сказати, що машина є «найбільш щасливим відгадувачем»; вона завжди вибирає перехід, який в кінцевому підсумку переводить її до кінцевого стану, якщо такий перехід

є.

Інший — уявити, що машина «розгалужується» на безліч копій, кожна з яких слідує за одним із можливих переходів. У той час як детермінована машина Тюрінґа має єдиний «шлях обчислень», за яким вона слідує, недетермінована - має «дерево обчислень». Якщо хоча б одна гілка дерева зупиняється з умовою «так», недетермінована машина Тюрінґа зупиняється.

2.2 Формальне визначення недетермінованої машини Тюрінґа

Недетерміновану машину Тюрінґа можна визначити як кортеж розміру шести:

- Γ – алфавіт недетермінованої машини Тюрінґа, скінченна множина символів, що можуть бути використані для запису на стрічку недетермінованої машини Тюрінґа;
- $b \in \Gamma$ - символ алфавіту Γ недетермінованої машини Тюрінґа, що позначає пустий символ;
- Q – непуста скінченна множина станів, у яких може перебувати машина Тюрінґа;
- $q_0 \in Q$ – елемент скінченної множини станів Q, що позначає початковий стан машини Тюрінґа;
- $F \subseteq Q$ – скінченна множина кінцевих станів, після переходу в які машина Тюрінґа зупиняється.
- $\delta \subseteq (Q \setminus F) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$ – багатозначне відображення на станах там символах алфавіту, що носить назву *відображення переходів*. L позначає перехід ліворуч R позначає перехід праворуч S позначає відсутність переходу

Стани та відображення переходів на станах, яке описує можливі дії машини Тюрінґа з урахуванням будь-якого можливого вмісту стрічки, такі ж, як і для стандартних детермінованих машин Тюрінґа, за винятком того, що відображення переходів більше не є однозначним. (Якщо машина Тюрінґа є детермінованою, то всі її можливі обчислення є префіксами одного, можливо,

нескінченного слова)

2.3 Зв’язок недетермінованої машини Тюрінґа з детермінованою

Будь-яку обчислювальну задачу, яку можна вирішити за допомогою детермінованої машини Тюрінґа, також можна вирішити за допомогою недетермінованої машини Тюрінґа, і навпаки. Однак вважається, що загалом часова складність може бути неоднаковою.

Детермінована машина Тюрінґа є частковим випадком детермінованої машини Тюрінґа, тому кожне обчислення, яке може виконуватися детермінованою машиною Тюрінґа, також може виконуватися еквівалентною недетермінованою машиною Тюрінґа.

Може здатися, що недетермінована машина Тюрінґа є потужнішою за детерміновану машину Тюрінґа, оскільки вони можуть дозволяти дерева можливих обчислень, що виникають із тієї ж початкової конфігурації, приймаючи рядок, якщо будь-яка гілка дерева приймає його. Проте можна недетерміновану машину Тюрінґа можна задати за допомоги детермінованої машини Тюрінґа, і насправді це можна зробити декількома способами.

Один підхід полягає у використанні детермінованої машини Тюрінґа, конфігурації якої представляють кілька конфігурацій недетермінованої машини Тюрінґа, а певна дія (операція) детермінованої машини Тюрінґа складається з відвідування кожної з них по черзі, виконання одного кроку під час кожного відвідування та створення нових конфігурацій, коли відображення переходу визначає кілька продовжень.

Інша конструкція моделює недетерміновану машину Тюрінґа з трьохстрічковими детермінованими машинами Тюрінґа, з яких перша стрічка завжди містить оригінальний вхідний рядок, друга використовується для моделювання конкретного обчислення недетерміновану машину Тюрінґа, а третя кодує шлях у дереві обчислень недетермінованої машини Тюрінґа[4]. Трьохстрічкові детерміновані машини Тюрінґа легко моделюються за допомогою звичайної машини Тюрінґа з однією стрічкою.

У другому підході сконструйована детермінована машина Тюрінґа ефективно виконує пошук у ширину в дереві обчислень недетермінованої машини Тюрінґа, відвідуючи всі можливі обчислення недетермінованої машини Тюрінґа в порядку збільшення довжини, поки не знайде прийнятне. Таким чином, довжина прийнятного обчислення детермінованої машини Тюрінґа, загалом, експоненціальна щодо довжини найкоротшого прийнятного обчислення недетермінованої машини Тюрінґа. Вважається, що це загальна властивість моделювання недетермінованої машини Тюрінґа за допомогою детермінованої машини Тюрінґа. Проблема P = NP, найвідоміше невирішене питання в інформатиці, стосується одного випадку цієї проблеми: чи кожна проблема, розв'язувана недетермінованою машиною Тюрінґа за поліноміальний час, обов'язково також розв'язується детермінованою машиною Тюрінґа за поліноміальний час.

2.4 Порівняння НДМТ з квантовими комп’ютерами

Оскільки квантові комп’ютери використовують квантові біти, які можуть бути в суперпозиціях станів, а не звичайні біти, іноді існує помилкова думка, що квантові комп’ютери є недетермінованими машинами Тюрінґа[5]. Однак експерти вважають (але це не було доведено), що потужність квантових комп’ютерів насправді непорівнянна з потужністю недетермінованих машин Тюрінґа; тобто, ймовірно, існують проблеми, які НДМТ може ефективно вирішити, які не може вирішити квантовий комп'ютер, і навпаки[6]. Зокрема, цілком імовірно, що NP-повні проблеми можна розв'язувати за допомогою НДМТ, але не квантових комп'ютерів за поліноміальний час.

2.5 Приклад НДМТ

Задамо машину Тюрінґа, що може записати на стрічку послідовність $S_0S_0S_0S_0S_0S_0$ ... , $S_1S_1S_1S_1S_1S_1$ або $S_0S_1S_1S_1S_1S_0$ ... тобто згенерувати під час роботи довільну стрічку, що містить символи S_0 та/або S_1

$$q_0S_0S_0Rq_1$$

$$q_0S_0S_1Rq_1$$

$$q_0 S_1 S_0 R q_1$$
$$q_0 S_1 S_1 R q_1$$
$$q_1 S_0 S_0 R q_2$$
$$q_1 S_0 S_1 R q_2$$
$$q_1 S_1 S_0 R q_2$$
$$q_1 S_1 S_1 R q_2$$
$$q_2 S_0 S_1 R q_0$$
$$q_1 S_0 S_0 R q_2$$
$$q_1 S_1 S_0 R q_2$$
$$q_2 S_1 S_1 R q_0$$

По правді кажучи, можна зменшити кількість станів до одного і залишити чотири інструкції:

$$q_0 S_0 S_0 R q_0$$
$$q_0 S_0 S_1 R q_0$$
$$q_0 S_1 S_0 R q_0$$
$$q_0 S_1 S_1 R q_0$$

У таблиці наведений варіант цієї НДМТ з трьома станами задля кращого розуміння побудови таблиць переходів, що описують функції переходу.

	$\boldsymbol{S_0}$	$\boldsymbol{S_1}$
$\boldsymbol{q_0}$	$S_0 R q_1, S_1 R q_1$	$S_0 R q_1, S_1 R q_1$
$\boldsymbol{q_1}$	$S_0 R q_2, S_1 R q_2$	$S_0 R q_2, S_1 R q_2$
$\boldsymbol{q_2}$	$S_1 R q_0, S_0 R q_0$	$S_1 R q_0, S_0 R q_0$

Табл. 2.5.1

РОЗДІЛ 3. УНІВЕРСАЛЬНА МАШИНА ТЮРІНҐА

3.1 Опис універсальної машини Тюрінґа

У інформатиці універсальна машина Тюрінґа — це машина Тюрінґа, яка моделює довільну машину Тюрінґа на довільному вході. Грубо кажучи, це інтерпретатор машин Тюрінґа. Принцип роботи УМТ такий: вона читає опис машини Тюрінґа, її вхідні дані. Даний принцип роботи було використано Джоном фон Нейманом у 1946 році для «Електронного обчислювального інструменту», який тепер носить назву фон Неймана: архітектура фон Неймана[7].

З точки зору обчислювальної складності, універсальна машина Тюрінґа з багатьма стрічками повинна бути повільнішою лише на логарифмічний коефіцієнт порівняно з машинами, котрі вона моделює[8].

Кожна машина Тюрінґа обчислює певну фіксовану часткову обчислювану функцію з вхідних рядків над своїм алфавітом. У цьому сенсі машина Тюрінґа поводиться як комп’ютер із фіксованою програмою. Однак ми можемо закодувати таблицю дій будь-якої машини Тюрінґа в рядок. Таким чином, ми можемо побудувати машину Тюрінґа, яка очікує на своїй стрічці рядок, що описує таблицю дій, за яким слідує рядок, що описує вхідну стрічку, і обчислює стрічку, яку обчислила б закодована машина Тюрінґа.

Завдяки такому кодуванню таблиць дій у вигляді рядків, в принципі, стає можливим для машин Тюрінґа відповідати на питання про поведінку інших машин Тюрінґа. Більшість із цих питань, однак, є невирішеними, а це означає, що розглянута функція не може бути обчислена механічно. Наприклад, проблема визначення того, чи зупиниться довільна машина Тюрінґа на певному вхідному сигналі чи на всіх входах, відома як проблема зупинки, була, загалом, нерозв’язною в оригінальній роботі Тюрінґа. Теорема Райса показує, що будь-яке нетривіальне питання про вихід машини Тюрінґа є нерозв'язаним.

Універсальна машина Тюрінґа може обчислити будь-яку рекурсивну

функцію, вирішити будь-яку рекурсивну мову та прийняти будь-яку рекурсивно перелічну мову. Згідно з тезою Черча-Тюрінґа, задачі, які можна розв'язувати універсальною машиною Тюрінґа, — це саме ті задачі, які можна розв'язувати за допомогою алгоритму або ефективного методу обчислень, для будь-якого розумного визначення цих термінів. З цих причин універсальна машина Тюрінґа служить стандартом для порівняння обчислювальних систем, а система, яка може моделювати універсальну машину Тюрінґа, називається повною Тюрінґом.

Абстрактною версією універсальної машини Тюрінґа є універсальна функція, обчислювана функція, яку можна використовувати для обчислення будь-якої іншої обчислюваної функції. Теорема про універсальну машину Тюрінґа доводить існування такої функції.

3.2 Ефективність УМТ

Без втрати загальності можна вважати, що вхід машини Тюрінґа міститься в алфавіті $\{0, 1\}$; будь-який інший кінцевий алфавіт може бути закодований через $\{0, 1\}$. Поведінка машини Тюрінґа M визначається її перехідною функцією. Цю функцію також можна легко закодувати як рядок над алфавітом $\{0, 1\}$. Розмір алфавіту машини Тюрінґа M, кількість стрічок, яку вона має, і розмір простору станів можна вивести з таблиці функції переходу. Розрізнені стани та символи можна ідентифікувати за їх положенням, наприклад, перші два стани за умовою можуть бути початковим і кінцевим станами.

Отже, кожна машина Тюрінґа може бути закодована як рядок над алфавітом $\{0, 1\}$. Крім того, ми вважаємо, що кожне неправильне кодування відображається на тривіальній машині Тюрінґа, яка негайно зупиняється, і що кожна машина Тюрінґа може мати нескінченну кількість кодувань, доповнюючи кодування довільною кількістю (скажімо) 1 в кінці. Не дивно, що ми можемо досягти такого кодування, враховуючи існування числа Геделя та обчислювальну еквівалентність між машинами Тюрінґа та μ-рекурсивними функціями. Подібним чином наша конструкція пов'язує з кожним двійковим рядком α машину Тюрінґа M_α.

Виходячи з наведеного вище кодування, у 1966 році Генні та Стернс показали, що для машини Тюрінґа M_α, яка зупиняється на введенні x протягом N кроків, існує універсальна машина Тюрінґа з багатьма стрічками, яка зупиняється на входах α, x (на різних стрічках) за $CN\ log\ N$, де C — константа, що характеризує машину, яка не залежить від довжини вхідного x, але залежить від розміру алфавіту машини Тюрінґа M, кількості стрічок та кількості станів.

Фактично це $O(Nlog\ N)$ моделювання, використовуючи нотацію Ландау. Відповідний результат для просторової складності, а не часової складності, полягає в тому, що ми можемо моделювати машину таким чином, що на будь-якому етапі обчислення використовується щонайбільше комірок CN, по-суті $O(N)$[9].

3.3 Маленькі УМТ

Коли Алан Тюрінґ придумав ідею універсальної машини, він мав на увазі найпростішу обчислювальну модель, достатньо потужну, щоб обчислити всі можливі функції, які можна обчислити. Клод Шеннон вперше чітко поставив питання про пошук найменшої можливої універсальної машини Тюрінґа в 1956 році. Він показав, що двох символів достатньо, якщо використовується достатньо станів (або навпаки), і що завжди можна обміняти стани на символи. Він також показав, що жодної універсальної машини Тюрінґа одного стану не може існувати.

У 1962 році Марвін Мінський відкрив 4-символьну універсальну машину Тюрінґа з 7 станами, використовуючи системи з 2 тегами. Якщо позначити (m, n) клас UTM з m станами та n символами, то знайдено такі кортежи: (15, 2), (9, 3), (6, 4), (5, 5), (4, 6), (3, 9) та (2, 18).

Однак узагальнення стандартної моделі машини Тюрінґа допускає навіть менші УМТ. Одним із таких узагальнень є можливість нескінченно повторюваного слова на одній або обох сторонах входу машини Тюрінґа, таким чином розширюючи визначення універсальності та відомої як «напівслабка» або «слабка» універсальність відповідно. Інші варіанти стандартної моделі машини

Тюринга, які дають невеликі універсальні машини Тюрінґа, включають машини з кількома стрічками або стрічками кількох розмірів, а також машини, поєднані з скінченним автоматом.

РОЗДІЛ 4. ІМОВІРНІСНА МАШИНА ТЮРІНҐА

4.1 Опис імовірнісної машини Тюрінґа

У теоретичній інформатиці ймовірнісна машина Тюрінґа — це недетермінована машина Тюрінґа, яка вибирає між доступними переходами в кожній точці програми відповідно до деякого розподілу ймовірностей. Як наслідок, ймовірнісна машина Тюрінґа може — на відміну від детермінованої машини Тюрінґа — мати стохастичні результати; тобто на заданому скінченному автоматі введення та інструкції він може мати різний час виконання або взагалі не зупинятися; крім того, він може прийняти ввід на вхід під час одного виконання і відхилити той самий вхід в іншому виконанні.

У випадку рівних імовірностей переходів імовірнісні машини Тюрінґа можна визначити як детерміновані машини Тюрінґа, що мають додаткову інструкцію «запис», де значення запису рівномірно розподілено в алфавіті машини Тюрінґа (загалом, однакова ймовірність запису «1» або «0» на стрічці). Інше поширене формулювання - це просто детермінована машина Тюрінґа з доданою стрічкою, повною випадкових клітин (можна порівняти з кубітами), яка називається "випадкова стрічка". Квантовий комп’ютер – це ще одна модель обчислень, яка за своєю суттю є ймовірнісною.

Імовірнісна машина Тюрінґа — це тип недетермінованої машини Тюрінґа, в якій кожен недетермінований крок є «підкиданням монети», тобто на кожному кроці є два можливих наступних ходу, і машина Тюрінґа ймовірно вибирає, який рух зробити[10].

4.2 Формальне визначення ІМТ

ІМТ – це кортеж $M = (Q, \sum, \Gamma, q_0, A, \delta_0, \delta_1)$:

- Q – скінченна множина станів ІМТ;
- $\sum -$ вхідний алфавіт;
- $\Gamma -$ алфавіт ІМТ, що включає в себе пустий символ;
- $q_0 \in Q$ – елемент множини станів Q, що позначає початковий стан машини ІМТ;

– $\delta_0 : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ – перша ймовірнісна функція переходу
– $\delta_1 : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ – перша ймовірнісна функція переходу

На кожному кроці ймовірнісна машина Тюрінґа застосовує або функцію переходу δ_0, або функцію переходу δ_1[11]. Цей вибір робиться незалежно від усіх попередніх виборів. Таким чином, процес вибору функції переходу на кожному кроці обчислення нагадує підкидання монети.

Імовірнісний вибір функції переходу на кожному кроці вносить помилку в машину Тюрінґа; тобто рядки, які має приймати машина Тюрінґа, можуть бути відхилені в деяких випадках, а рядки, які машина Тюрінґа має відхилити, можуть бути прийняті в деяких випадках. Для цього кажуть, що мова L розпізнається з ймовірністю помилки e ймовірнісною машиною Тюрінґа M, якщо:

– стрічка w є у мові L означає, що $\mathrm{P}(M \text{ приймає } w) \geq 1 - e$
– стрічки w немає у мові L означає, що $\mathrm{P}(M \text{ відхиляє } w) \geq 1 - e$

4.3 Класи складності обчислень ІМТ

Теорія складності обчислень моделює рандомізовані алгоритми як імовірнісні машини Тюрінґа. Основним рандомізованим класом складності є RP, який є класом задач проблеми вибору, для яких існує ефективний (за поліноміальним часом) рандомізований алгоритм (або імовірнісна машина Тюрінґа), який з абсолютною впевненістю розпізнає екземпляри НІ та з імовірністю розпізнає екземпляри ТАК. щонайменше 1/2. Класом доповнення для RP є co-RP. Класи задач, які мають алгоритми з поліноміальним середнім часом виконання, вихідні дані яких завжди правильні, називаються ZPP.

У результаті помилки, яка виникає внаслідок імовірнісного підкидання монети, поняття прийняття рядка імовірнісною машиною Тюрінґа може бути визначено різними способами. Одне з таких понять, яке включає кілька важливих класів складності, передбачає ймовірність помилки 1/3. Наприклад, клас складності BPP визначається як клас мов, розпізнаних імовірнісною машиною

Тюрінґа за поліноміальний час з імовірністю помилки 1/3. Іншим класом, визначеним за допомогою цього поняття прийняття, є BPL, який є тим самим, що і BPP, але накладає додаткове обмеження, що мови повинні бути розв’язними в логарифмічному просторі.

Класи складності, що випливають з інших визначень прийняття стрічки, включають RP, co-RP і ZPP.

Одним із центральних питань теорії складності є те, чи додає випадковість потужності; тобто чи існує проблема, яку можна вирішити за поліноміальний час імовірнісною машиною Тюрінґа, але не детермінованою машиною Тюрінґа? Або чи можуть детерміновані машини Тюрінґа ефективно моделювати всі імовірнісні машини Тюрінґа з щонайбільше поліноміальним уповільненням? Відомо, що $P \subseteq BPP$, оскільки детермінована машина Тюрінґа є лише окремим випадком імовірнісної машини Тюрінґа. Однак невідомо, чи є $BPP \subseteq P$ (а це означає, що $BPP = P$). Те саме питання щодо логарифмічного простору замість поліноміального часу (чи $L = BPLP$?).

РОЗДІЛ 5. РОЗРОБКА ОБ'ЄКТУ ДОСЛІДЖЕНЬ

5.1 Короткі висновки про машини Тюрінґа

Однією з найголовніших цілей цієї випускної кваліфікаційної бакалаврської роботи є створення інтерпретатора до машин Тюрінґа. Для того щоб це зробити, треба використати знання про машини Тюрінґа, які я наводив у попередніх розділах.

Основна мета опису різних машин Тюрінґа – це показати, що їх, по-перше, дуже багато. По-друге, більшість з них можна звести до традиційної машини Тюрінґа – тобто детермінованої машини Тюрінґа – яку Алан Тюрінґ описав у своїй докторській роботі.

Який висновок з цього можна зробити? Необов'язково імплементовувати кожний варіант машини Тюрінґа – головне знати, як її потім пристосувати до того інтерпретатора машини Тюрінґа, що був розроблений, а саме до ДМТ. Наприклад, для недетермінованої машини Тюрінґа на кожному кроці, що має більше одного переходу, можна робити копію екземпляра машини Тюрінґа.

5.2 Огляд підходів візуалізації

Існує багато способів візуалізації певних процесів, головні з них – це використання графічного та текстового інтерфейсу користувача. Оскільки задача кваліфікаційної роботи – це розробка інтерпретатора машини Тюрінґа для використання у навчанні (навчальна програма), то без таких підходів візуалізації тут не обійтись.

Виникає логічне питання – а який спосіб візуалізації краще вибрати? Тут треба відштовхуватись від двох факторів, а саме: сфера застосування та поточний стан цифрових технологій в Україні. По-перше, так як сфера застосування – це галузь освіти, то було б логічно використати той спосіб, який би зменшував складність використання програми та був цікавим. По-друге, згідно з дослідженням, що проводили Media Landscapes, шістдесят вісім відсотків телефонів, які мають українці, є смартфонами. У більшості сімей в

Україні є комп’ютери, і водночас більшості закладах вищої освіти є комп’ютерне обладнання.

Виходячи з такого стану речей, використання графічного інтерфейсу є виправданим, бо графічний інтерфейс є більш зрозумілим для користувача, аніж текстовий інтерфейс, а сам користувач має доступ до тих технологій, що дозволяють використання такого ПЗ з графічним інтерфейсом. Слід зазначити, що використання програм з текстовим інтерфейсом потребує значно менше пам’яті, але потужність сучасних (і навіть не зовсім сучасних) пристроїв нівелює такий недолік графічного інтерфейсу.

5.3 Вибір технології та мови програмування

Розгляд технологій (платформ), що потенційно можуть бути вибрані для реалізації об’єкту розробки, повинен бути ретельним. Від цього залежить якість продукту, його життєздатність та можливість бути впровадженим у систему освіти. Треба зауважити, що вибір технології прямо впливає на підтримання продукту, на можливість його доопрацювання або модифікації.

Потрібно визначити критерії, яким повинна відповідати технологія. По-перше, спираючись на вищенаведені факти про сучасний стан цифрових технологій в Україні, продукт повинен бути здатним працювати на сучасних приладах, у тому числі й на мобільних пристроях.

По-друге, продукт має бути написаний високорівневою мовою програмування. Це є запорукою того, що програма може бути легко модифікована або її частина може бути використана для написання нової навчальної програми.

З огляду на критерії, вебзастосунок – є одним із найбільш вдалим варіантом для вибору. По-перше, використання такого застосунку є можливим як на комп’ютерах, так і на мобільних пристроях. По-друге, у вебзастосунках використовуються мови програмування високого рівня.

5.4 Встановлення засобів для розробки об’єкту досліджень

Створення вебзастосунку буде проводитись з використанням мови JavaScript (TypeScript). Для того, щоб запускати код на JavaScript, потрібен рушій V8. Він є зараз у кожному сучасному браузері.

V8 — це безкоштовний рушій JavaScript з відкритим кодом, розроблений Chromium Project для вебпереглядачів Google Chrome і Chromium. Творцем проєкту є Ларс Бак. Перша версія рушія V8 була випущена одночасно з першою версією Chrome: 2 вересня 2008 року. Він також використовувався на стороні сервера, наприклад, у Couchbase та Node.js.

Сервер для вебзастостосунку буде працювати у середовищі Node.js. Node.js — це міжплатформне середовище виконання мови програмування JavaScript із відкритим вихідним кодом, яке працює на рушії V8 і виконує код JavaScript за межами вебпереглядача. Node.js дозволяє розробникам використовувати мову програмування JavaScript для написання інструментів командного рядка та для сценаріїв на стороні сервера — запуск сценаріїв на стороні сервера для створення динамічного вмісту вебсторінки, перш ніж сторінка буде відправлена у веббраузер користувача. Отже, Node.js являє собою парадигму «JavaScript всюди», що об’єднує розробку вебзастосунків навколо однієї мови програмування, а не різних мов для сценаріїв на стороні сервера та клієнта.

Node.js має архітектуру, керовану подіями, з можливістю асинхронного введення-виведення. Ці варіанти дизайну спрямовані на оптимізацію пропускної здатності та масштабованості у вебзастосунках з багатьма операціями введення/виводу, а також для вебзастосунків, що працюють у реальному часі (наприклад, комунікаційні програми в режимі реального часу та браузерні ігри тощо).

Проєкт розподіленої розробки Node.js раніше керувався Node.js Foundation, а тепер об’єднався з JS Foundation, щоб утворити OpenJS Foundation, якому сприяє програма Collaborative Projects Linux Foundation.

Корпоративні користувачі програмного забезпечення Node.js включають GoDaddy, Groupon, IBM, LinkedIn, Microsoft, Netflix, PayPal, Rakuten, SAP, Walmart, Yahoo!, і Amazon Web Services. Node.js можна завантажити з сайту https://nodejs.org/uk/download/.

5.5 Опис основних сутностей

TypeScript дозволяє писати код у парадигмі ООП. Визначмо кілька основних класів, що допоможуть описати певні сутності машини Тюрінґа.

5.5.1 Опис сутності машини Тюрінґа

Створімо клас TuringMachine, що буде інкапсулювати дані й відтворювати логіку машини Тюрінґа.

```
export class TuringMachine {
    readonly stateMachine: StateMachine;

    readonly tape: Tape;

    private currentStep: number;

    private state: TuringMachineState;

    constructor() {


        this.stateMachine = new StateMachine();
        this.tape = new Tape();

        this.currentStep = 0;
        this.state = TuringMachineState.READY;
    }
```

Рисунок 5.5.1.1 – клас TuringMachine

Розглянемо цей клас. У ньому зберігається багато речей:

По-перше, це внутрішній стан, у якому перебуває машина Тюрінґа. Тут головне не плутати стани, у яких знаходиться машина Тюрінґа під час зчитування стрічки – це стани, які визначені формально, для обчислення

абстрактною машиною програми машини Тюрінґа – із внутрішніми станами, які зберігають інформацію про те, чи йдуть обчислення машиною Тюрінґа, чи вона зупинилась.

Для зручності у програмі визначені такі внутрішні стани:

- машина готова працювати
- машина працює
- машина зупинилась у кінцевому стані
- машина зупинилась з інших причин (через помилку)

У коді сутності внутрішнього стану машини Тюрінґа відповідає перелічуваний тип даних TuringMachineState:

```
export enum TuringMachineState {
    READY = "Ready",
    RUNNING = "Running",
    STOPPED = "Stopped",
    FAULTY = "Faulty"
}
```

Рисунок 5.5.1.2 – enum TuringMachineState

По-друге, у класі TuringMachine зберігається кількість кроків, що були виконані машиною Тюрінґа. Яка практична користь від цього? Насправді, не кожна машина Тюрінґа колись зупиниться, це обґрунтовується проблемою зупинки. Тому в програмі можна задати максимальне число кроків, що може виконати машина Тюрінґа.

5.5.2 Опис сутності стрічки МТ

У класі TuringMachine зберігається елемент стрічки класу Tape, що відповідає сутності стрічки машини Тюрінґа. Він виконує одночасно роль як і стрічки машини Тюрінґа, так і головки цієї стрічки.

```
export class Tape {
    private static readonly EMPTY_CELL_FILLING_CHAR: string = " ";

    private content: TapeSymbol[];

    private headPosition: number;


    constructor(content: TapeSymbol[] = []) {
        this.content = content;
        this.resetHeadPosition();
    }
```

Рисунок 5.5.2.1 – клас Tape

У класі Tape зберігається символів алфавіту машини Тюрінґа, який зберігає вміст стрічки машини Тюрінґа. У конструкторі класу Tape ми бачимо, що цей масив ініціалізується, а головка стрічки переходить на найпершу клітну стрічки. Якщо конструктор викликається без параметрів, то вміст стрічки ініціалізується даними за замовчуванням, а саме: вміст стрічки буде складатись із пустих символів.

Константою зберігається пустий символ, програмно це реалізовано як UTF16-код 0x20, тобто через пробіл. Також зберігається номер поточної клітинки, у якій знаходиться машина Тюрінґа. Це потрібно для того, щоб зберігти логіко роботи зі стрічкою у самій стрічці.

У класі Tape реалізована можливість експортувати зміст стрічки, це робить зручним поширення програми машини Тюрінґа іншим користувачам.

5.5.3 Опис сутності стану і переходу

Станами у машині Тюрінґа оперує машина станів. Машина станів представлена у програмі класом StateMachine. Цей клас відповідає керування станами машини Тюрінґа. Цей клас зберігає початковий стан та масив усіх інших станів. Клас також зберігає переходи між станами. Перехід представлений класом Transition.

```
export class State {
    private static nextStateID: StateID = 1;

    readonly id: StateID;

    private position: Position;

    private label: string;

    private final: boolean;

    private inTransitions: Map<TransitionID, Transition>;

    private outTransitions: Map<TransitionID, Transition>;

    private symbolsToOutTransitions: Map<TapeSymbol, Set<Transition>>;

    constructor(position: Position, label: string, final: boolean = false) {
        this.id = State.nextStateID;
        State.nextStateID++;

        this.position = position;
        this.label = label;
        this.final = final;

        this.inTransitions = new Map();
        this.outTransitions = new Map();
        this.symbolsToOutTransitions = new Map();
    }
```

```
export class Transition {
    private static nextTransitionID: TransitionID = 1;

    readonly id: TransitionID;

    readonly origin: State;

    readonly destination: State;

    private inputSymbol: TapeSymbol;

    private outputSymbol: TapeSymbol;

    private headAction: HeadAction;

    constructor(origin: State,
        destination: State,
        inputSymbol: TapeSymbol,
        outputSymbol: TapeSymbol,
        headAction: HeadAction) {
        this.id = Transition.nextTransitionID;
        Transition.nextTransitionID++;

        this.origin = origin;
        this.destination = destination;

        this.inputSymbol = inputSymbol;
        this.outputSymbol = outputSymbol;
        this.headAction = headAction;
    }
```

Рисунок 5.5.2.1 – класи State і Transition

У класі State зберігається позиція елемента у візуальному 2D просторі; переходи, що ведуть до цього стану та з цього стану, назва цього стану; символи, що можна прочитати у цьому стані і записати у комірку, перебуваючи в цьому конкретному стані та дані про те, чи є цей стан кінцевим.

У класі Transition є посилання на стан, з якого відбувається перехід, та стан у який відбувається перехід. Також там зберігається символ, який треба записати перед переходом, та символ, який був прочитаний. Окрім цього, у класі визначено напрямок руху по стрічці, що виконується після переходу.

5.6 Огляд інтерфейсу програми

Так виглядає інтерфейс програми:

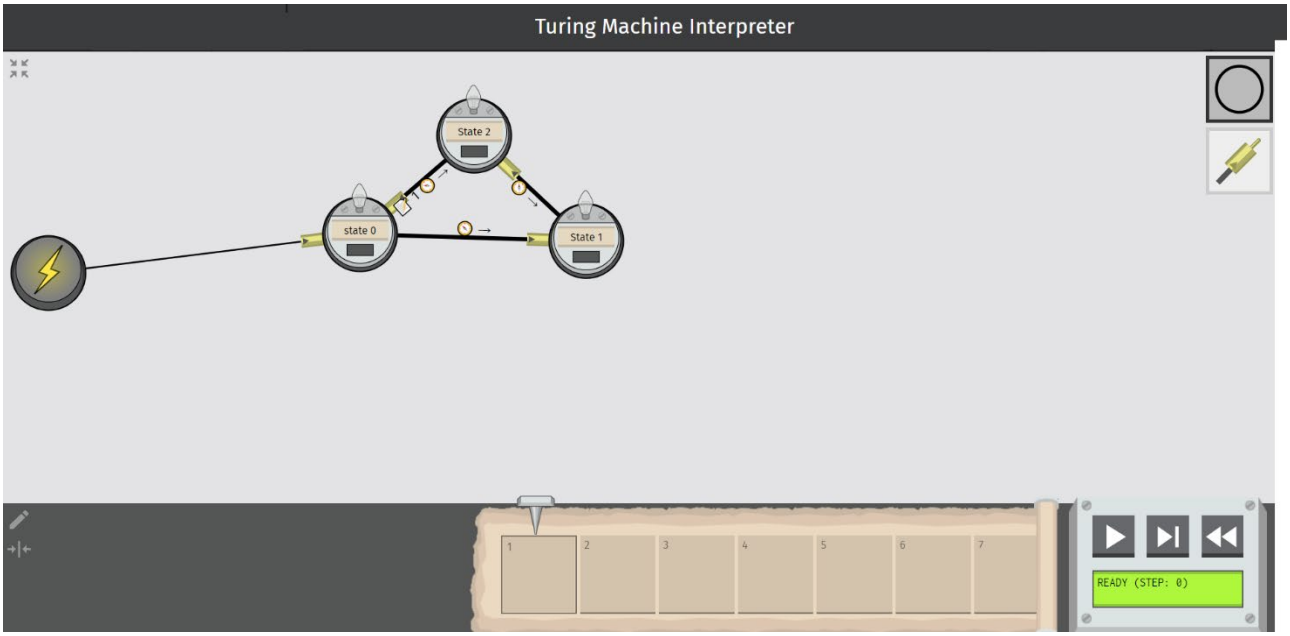

Рисунок 5.6.1 – інтерфейс програми

На рисунку 5.6.1 можна побачити три стани, переходи між ними, стрічку, меню керування станами і переходами, і меню для керування виконанням машини Тюрінґа.

У меню виконання є кнопки: виконати машину Тюрінґа (як уже було зазначено, у коді можна визначати максимальну кількість кроків, які можуть бути виконані машиною Тюрінґа), виконати крок і кнопка для очищення стрічки і повернення машини до початкового стану.

Праворуч можна побачити меню керування станами і переходами. Коли активна кнопка станів, то стани створюється натисканням лівої кнопки мишки. Якщо натиснути на стан, можна змінити його назву та зазначити, чи є цей стан кінцевим. Коли кнопка переходів активна, то можна створювати переходи: треба навести мишку на один зі станів, затиснути ліву кнопку мишку кнопку і відвести мишку на інший стан, тоді відкриється меню переходів між цими двома станами. Початковим станом є той, до якого йде перехід від іконки із блискавкою.

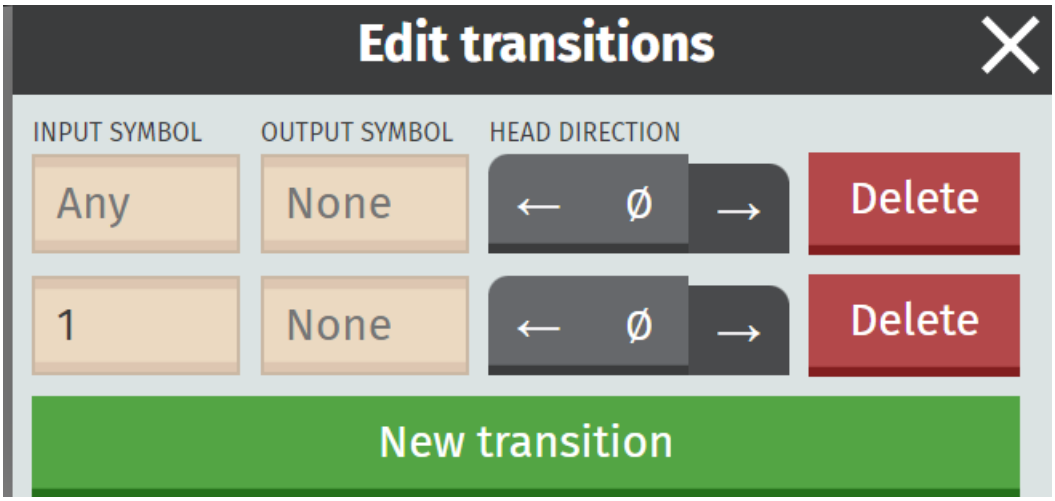

Рисунок 5.6.2 – меню редагування переходів між станами

Вміст стрічки можна змінювати. Для цього треба натиснути на будь-яку клітину стрічки.

5.7 Приклад роботи інтерпретатора машин Тюрінґа

Розгляньмо машину Тюрінґа задану у Таблиці 1.2.1. Вона повинна записувати на стрічку послідовність $S_0S_0S_1S_0S_0S_1$ … , де S_0 – пустий символ. По-перше, треба перенести ці стани до програми. Скористаймося графічним інтерфейсом для цього:

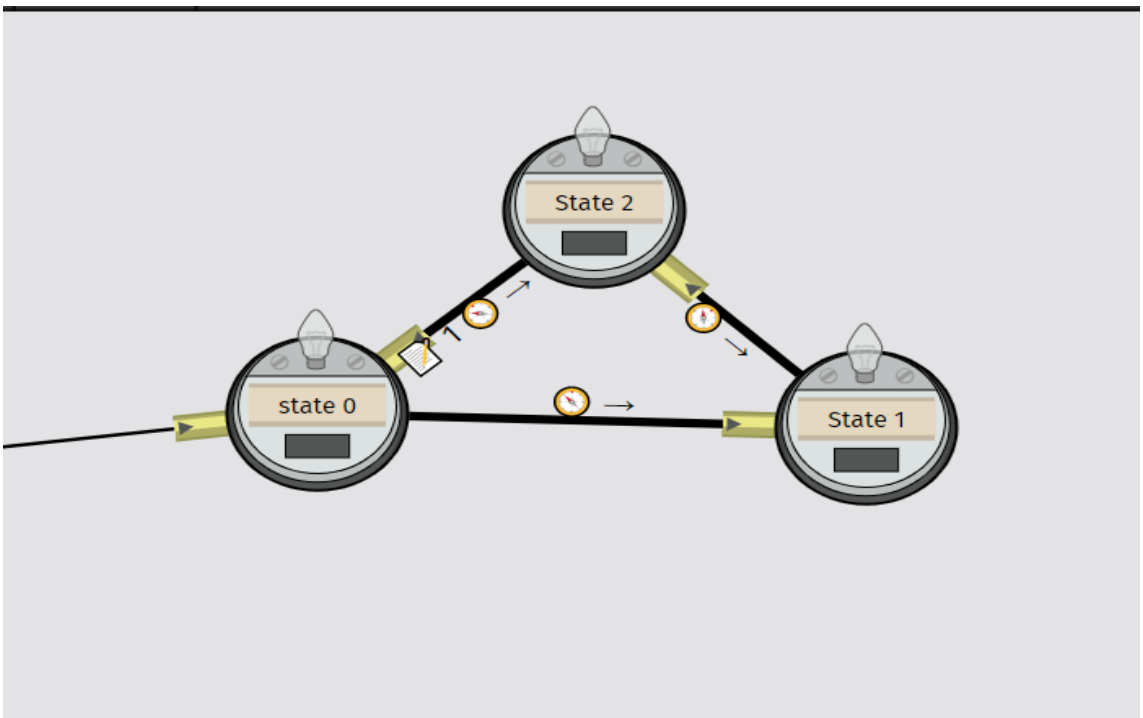

Рисунок 5.7.1 – графічне представлення станів

{"stateMachine":{"states":[{"id":1,"position":{"x":
562.4000244140625,"y":100.20001220703125},"label":"State
2","final":false},{"id":2,"position":{"x":696.0000610351562,"y":
225.80001831054688},"label":"State 1","final":false},{"id":
3,"position":{"x":426.4000244140625,"y":
217.80001831054688},"label":"state 0","final":false}],"transitions":
[{"id":1,"originID":3,"destinationID":
2,"inputSymbol":"","outputSymbol":"","headAction":1},{"id":
2,"originID":3,"destinationID":
2,"inputSymbol":"1","outputSymbol":"","headAction":1},{"id":
3,"originID":2,"destinationID":
1,"inputSymbol":"","outputSymbol":"","headAction":1},{"id":
4,"originID":2,"destinationID":
1,"inputSymbol":"1","outputSymbol":"","headAction":1},{"id":
5,"originID":1,"destinationID":
3,"inputSymbol":"","outputSymbol":"1","headAction":1},{"id":
6,"originID":1,"destinationID":
3,"inputSymbol":"1","outputSymbol":"1","headAction":
1}],"initialStateID":3},"tape":{"content":[]}}

Рисунок 5.7.2 – представлення станів у Json

Запустимо дану програму на пустій стрічці.

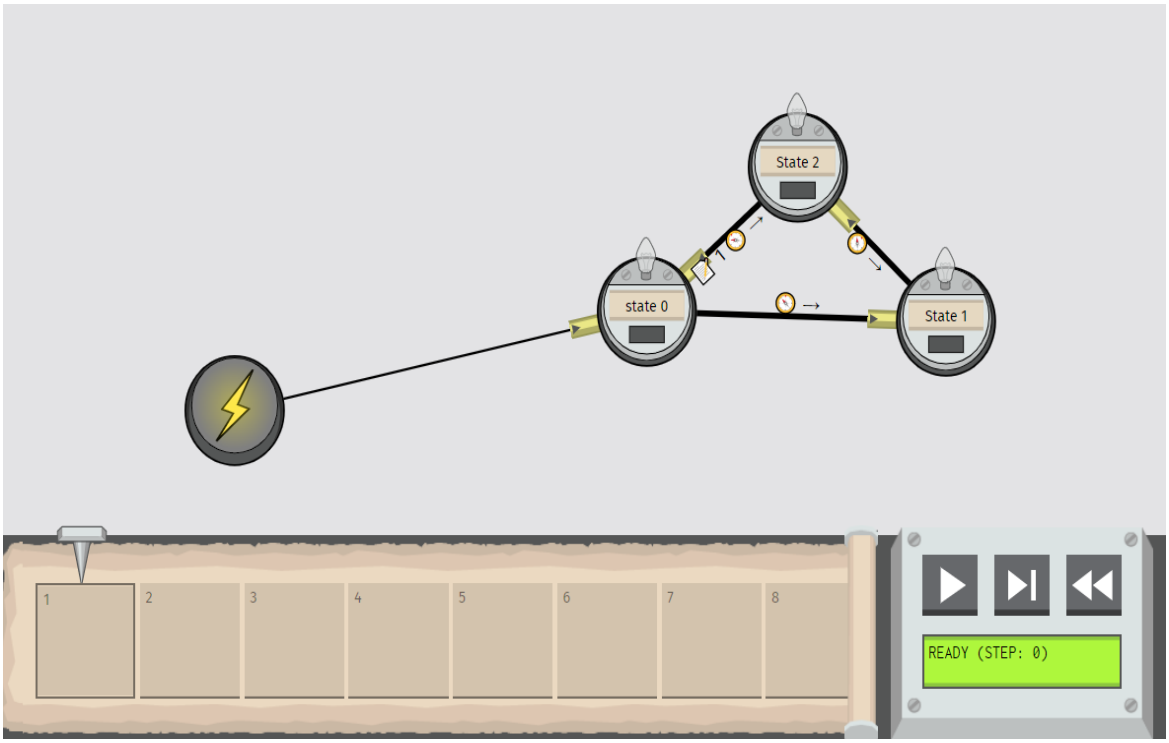

Рисунок 5.7.3 – нульовий крок

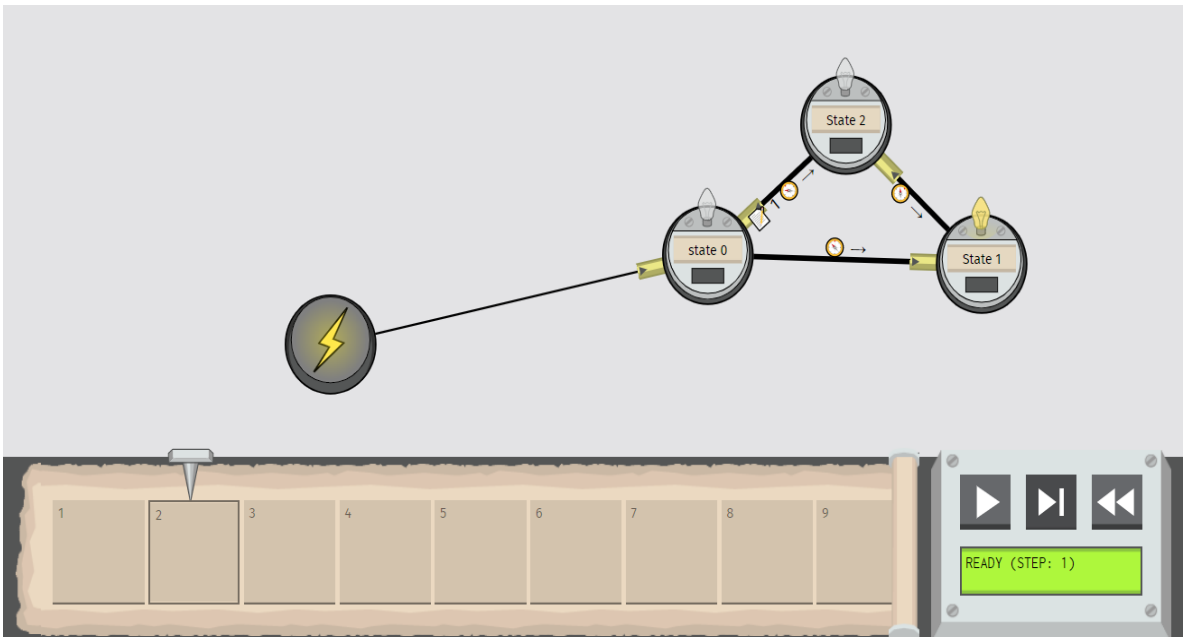

Рисунок 5.7.4 – перший крок

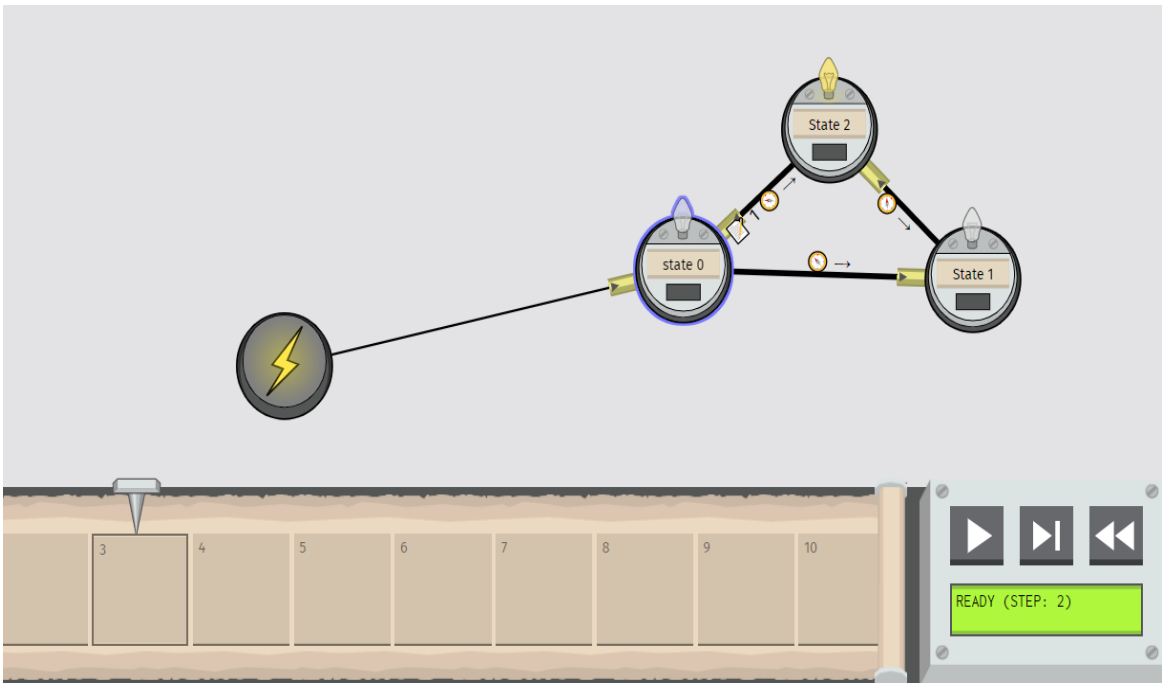

Рисунок 5.7.5 – другий крок

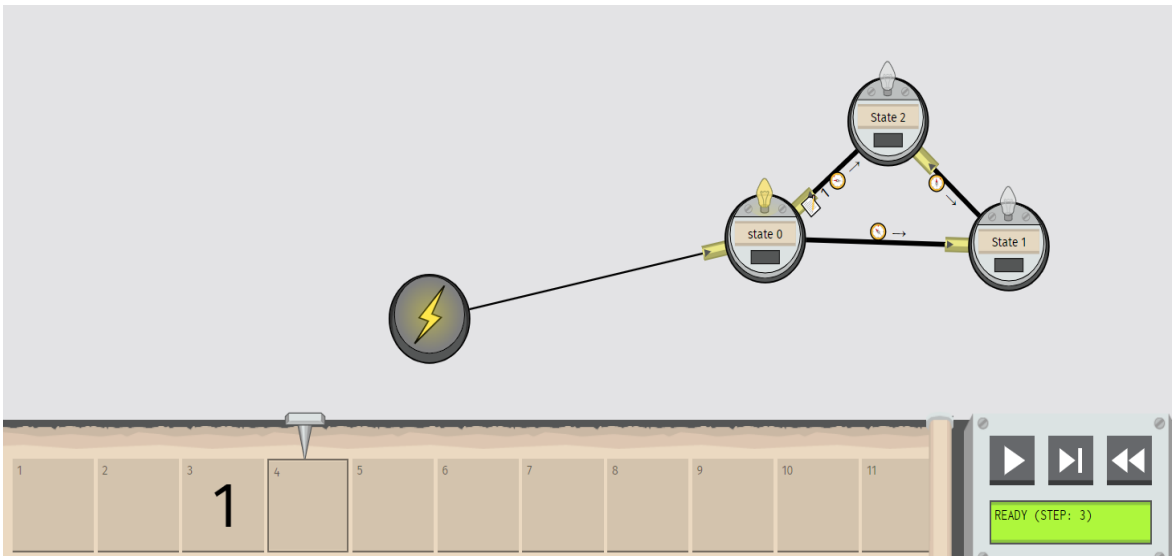

Рисунок 5.7.6 – третій крок

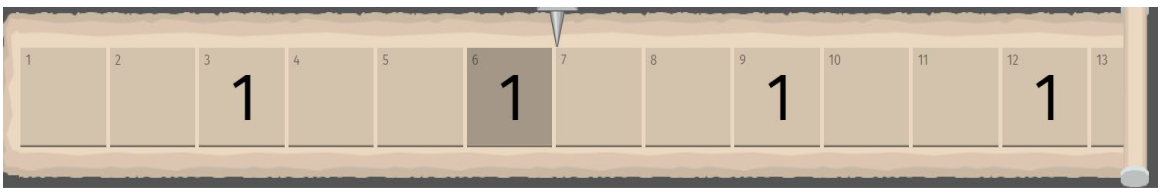

Рисунок 5.7.7 – стрічка після виконання 13 кроків

Машина не має кінцевого стану та буде працювати без зупинки.

5.8 Технічні характеристики ПК

Програма виконувалась на ПК з процесором Intel Core i3-7130u з тактовою частотою 2.70 ГГц та 4 логічними ядрами; з оперативною пам’яттю об’ємом 8 Гб.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було розроблено інтерпретатор машин Тюрінґа, досліджено різні варіанти машин Тюрінґа та спосіб звести деякі з них до детермінованої машини Тюрінґа, інтерпретатор до якої був розроблений.

У Розділі 5 було докладно розглянуто процес виконання розробки інтерпретатора: робота виконувалась за допомогою редактору коду Visual Studio Code, у середовищі Node.js. Були описані основні сутності машини Тюрінґа та відповідні класи, що використовуються у програмі.

Була показана робота програми на прикладі. Виконання програми пройшло успішно.

Програму можна використовувати для здобуття навичок та знань про машину Тюрінґа у закладах вищої освіти. Програма може швидко розгортатись на серверах локальних мереж вищих навчальних закладів, або може бути завантаженою окремо студентом (здобувачем освіти) і запущеною без використання сервера.

Таким чином, було розроблено інтерпретатор машини Тюрінґа (начальну програму), що може бути успішно використовуватись вже сьогодні.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. David Hilbert and Wilhelm Ackermann. Grundzüge der Theoretischen Logik. Springer, Berlin, Germany, 1928.
2. Hodges, Andrew (2012). Alan Turing: The Enigma (The Centenary ed.). Princeton University Press. ISBN 978-0-691-15564-7.
3. Turing 1936 in The Undecidable 1965:145
4. Lewis, Harry R.; Papadimitriou, Christos (1981). "Section 4.6: Nondeterministic Turing machines". Elements of the Theory of Computation (1st ed.). Englewood Cliffs, New Jersey: Prentice-Hall. pp. 204–211. ISBN 978-0132624787.
5. The Orion Quantum Computer Anti-Hype FAQ.
6. Tušarová, Tereza (2004). "Quantum complexity classes".
7. Martin Davis, The universal computer : the road from Leibniz to Turing (2017).
8. Arora and Barak, 2009, Theorem 1.9.
9. Arora and Barak, 2009, Exercises 4.1
10. Sipser, Michael (2006). Introduction to the Theory of Computation (2nd ed.). USA: Thomson Course Technology. p. 368. ISBN 978-0-534-95097-2.
11. Arora, Sanjeev; Barak, Boaz (2016). Computational Complexity: A Modern Approach. Cambridge University Press. p. 125. ISBN 978-0-521-42426-4.