МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем

(повна назва кафедри (предметної, циклової комісії))

Дипломна робота

на здобуття ступеня вищої освіти «бакалавр»

(освітньо-кваліфікаційний рівень)

на тему «Програмна система перетворення моделей алгоритмів»
«Software for Conversion of Algorithm Models»

Виконала: студентка денної форми навчання
спеціальності 123 Комп'ютерна інженерія

(шифр і назва напряму підготовки, спеціальності)

Воронич Марія Сергіївна

(прізвище, ім'я, по-батькові)

Керівник ст. викл. Лісіцина І. М.

(науковий ступінь, вчене звання, прізвище та ініціали, **підпис**)

Рецензент к.ф.-м.н., доц. Петрушина Т. І.

(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№___ від «___»_____ 2020 р.

Завідувач кафедри

_____ _Є.В. Малахов_

(підпис) (прізвище, ініціали)

Захищено на засіданні ЕК № ___

протокол № __ від «___» ____ 2020 р.

Оцінка _____/_____/_____

(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

_____ _Н.Ф. Казакова_

(підпис) (прізвище, ініціали)

Одеса - 2020

АНОТАЦІЯ

Мета роботи – розробка інструменту, що виконує перетворення моделей алгоритмів для використання в якості підтримуючого програмного забезпечення при вивченні регулярних виразів, теорії алгоритмів, теорії програмування, прикладних аспектів конструювання компіляторів і т. д.

В результаті цієї роботи розроблено програмну систему, що виконує перетворення регулярних виразів у скінченні автомати та перетворення скінчених автоматів; візуалізує отримані результати у вигляді діаграм переходів. Система може бути використана в якості підтримуючого програмного забезпечення в дисциплінах, пов'язаних з розробкою компіляторів, теорією алгоритмів, розробкою інформаційно-пошукових систем.

ABSTRACT

The aim of the thesis is the development of a tool that performs the conversion of algorithm models and can be used as supporting software for studying regular expressions, algorithm theory, programming theory, applied aspects of compiler construction, etc.

The result of the work performed is software simulating conversion of a regular expression to finite automata and conversion of finite automata; visualizes converted finite automata as transition diagrams. The system can be used as supporting software for studying subjects related to compiler development, algorithm theory, development of information retrieval systems.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

ДСА — детермінований скінчений автомат

НСА — недетермінований скінчений автомат

ВСТУП

У теорії і практиці програмування вже протягом десятиліть використовуються такі моделі алгоритмів як регулярні мови і скінчені автомати [1].

Регулярна мова — це найбільш простий та розповсюджений клас формальних мов, тобто, математичних моделей реальної мови, де під реальною мовою розуміється деякий спосіб комунікації суб’єктів один з одним. Для визначення регулярних мов найчастіше використовуються регулярні граматики, регулярні вирази та скінчені автомати.

З точки зору сучасної практики програмування великий інтерес представляють регулярні вирази. Аналіз текстової інформації, фільтрація великих масивів даних, розробка компонентів компілятора [2], валідація даних користувачів, перевірка відповідей у пошукових системах, системах тестування знань [3] й т. д. — це прикладні задачі, що вирішуються даним інструментом.

Розпізнавачами для регулярних виразів є скінчені автомати. Реальні програмні продукти, наприклад, утиліта grep у UNIX–подібній системі та застосунки (текстові редактори ed, sed, vim, Notepad++ і т. д.) компілюють регулярний вираз у детермінований або недетермінований скінчені автомати, які згодом видозмінюються для розпізнавання патернів в тексті, що оброблюється. Генератори лексичних аналізаторів, такі як Lex і Flex, будучи компонентами компілятора, отримують формальні описи лексем, по суті — регулярних виразів, і генерують детермінований скінчений автомат, що розпізнає, яка з лексем з'являється на його вході. Причиною такого рішення було те, що в разі, коли виникає необхідність відредагувати лексичний аналізатор, то набагато простіше і безпечніше внести зміни в регулярний вираз, ніж витрачати час на відладку коду, щоб виправити виниклий дефект [4].

Всі популярні мови програмування включають бібліотеки підтримки цього інструменту або навіть мають реалізацію цієї підтримки, вбудованої безпосередньо в саму мову. У якості відомих прикладів виступають мови Perl, Java, JavaScript, C#, Ruby, Python і багато інших [5].

Незважаючи на те, що регулярні вирази алгебраїчно задають такі ж мови, що і скінчені автомати — регулярні мови — існує явна різниця між цими двома моделями, яка полягає в тому, що регулярні вирази визначають допустимі послідовності символів декларативним способом [4].

Побудова регулярного виразу є нетривіальним завданням. Вираз може бути громіздким і невиправдано складним. Не всі наважуються використовувати регулярні вирази через складність їх освоєння [3].

Метою дипломної роботи є розробка інструменту, що виконує перетворення моделей алгоритмів, який можна використовувати в якості підтримуючого програмного забезпечення при вивченні власне регулярних виразів, а також теорії алгоритмів, теорії програмування, прикладних аспектів конструювання компіляторів і т. д. Цей інструмент дозволить, наприклад, наочно відображати результат перетворення регулярного виразу в скінчений автомат у вигляді діаграми переходів. Для отримання такої діаграми можливо використовувати наступні методи:

1) алгоритм Томпсона: перетворення регулярного виразу у НСА;
2) алгоритм побудови підмножини: перетворення НСА у ДСА;
3) алгоритм Хопкрофта: мінімізація ДСА.

Інструмент повинен надавати можливість візуалізації отриманих результатів у вигляді графів (тобто діаграм переходів).

ВИСНОВКИ

Результатом даної роботи є програмна система, що симулює процеси перетворення регулярних виразів у скінченні автомати, візуалізує отримані результати у вигляді діаграм переходів. Система може бути використана в якості підтримуючого програмного забезпечення в навчальних дисциплінах, пов'язаних з розробкою компіляторів, теорією алгоритмів, розробкою інформаційно-пошукових систем.

Розроблений програмний продукт може бути корисний при побудові систем автоматизованого контролю знань з питаннями у відкритій формі, в яких тестований повинен скласти відповідь у вигляді рядка, а не вибирати його із запропонованих варіантів. У цьому випадку можливе значна кількість варіантів правильної відповіді. Рішення проблеми перевірки таких тестів полягає в використанні для перевірки регулярних виразів.

Ще одним прикладом застосування є автоматична перевірка виконання завдань на побудову регулярних виразів.

У майбутніх версіях система може бути доповнена іншими алгоритмами для обраних моделей, і іншими моделями алгоритмів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Поляков В. И. Преобразование моделей алгоритмов / В. И. Поляков, В. И. Скорубский. // ИЗВ. ВУЗОВ. ПРИБОРОСТРОЕНИЕ. – 2012. – С. 41–46.
2. Компиляторы: принципы, технологии и инструментарий / А.Ахо, С. Моника, Р. Сети, Р. Ульман. – Москва: Вильямс, 2008. – 1184 с. – (2-е изд.).
3. Сычев О. А. Инструменты помощи автору регулярных выражений для тестовых вопросов в СДО Moodle / О. А. Сычев, Г. В. Терехов. // Открытое образование. – 2016. – С. 43–50.
4. Хопкрофт Д. Введение в теорию автоматов, языков и вычислений / Д. Хопкрофт, Р. Мотвани, Д. Ульман., 2008. – 528 с. – (2-е изд.).
5. Фицджеральд М. Регулярные выражения: основы / М. Фицджеральд., 2015. – 144 с.
6. Братчиков И. Л. Синтаксис языков программирования / И. Л. Братчиков. – Москва: Наука, 1975. – 232 с. – (Главная редакция физико-математической литературы).
7. Sipser M. Introduction to the Theory of Computation / Michael Sipser. – Boston,: Cengage Learning, 2013. – 458 с. – (Third Edition).
8. Ахо А. Теория синтаксического анализа, перевода и компиляции / А. Ахо, Д. Ульман. – Москва: Мир, 1978. – 612 с. – (Том 1: Синтаксический анализ).
9. Regexper [Електронний ресурс] – Режим доступу до ресурсу: https://regexper.com/.
10. Debuggex [Електронний ресурс] – Режим доступу до ресурсу: https://www.debuggex.com/.
11. FSM simulator [Електронний ресурс] – Режим доступу до ресурсу: http://ivanzuzak.info/noam/webapps/fsm_simulator/.

12. ExtendsClass: Regex Tester [Електронний ресурс] – Режим доступу: https://extendsclass.com/regex-tester.html.
13. CyberZHG's Toolbox [Електронний ресурс] – Режим доступу: https://cyberzhg.github.io/toolbox/.
14. Python — Википедия [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Python.
15. Python 2 vs Python 3: Key Differences [Електронний ресурс] – Режим доступу: https://www.guru99.com/python-2-vs-python-3.html.
16. Graphviz - Wikipedia [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Graphviz.
17. Graphviz - Graph Visualization Software [Електронний ресурс] – Режим доступу: https://www.graphviz.org/.
18. User Guide — graphviz 0.14 documentation [Електронний ресурс] – Режим доступу: https://graphviz.readthedocs.io/en/stable/manual.html.
19. Visual Studio Code - Wikipedia [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Visual_Studio_Code.
20. Convert Infix to Postfix Expression [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/Convert-Infix-to-Postfix-Expression#:~:text=To%20convert%20infix%20expression%20to,maintaining%20the%20precedence%20of%20them.