BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC CẦN THƠ**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

**□  □**



**BÁO CÁO**

**TIN HỌC LÝ THUYẾT (CT12102)**

**Đề tài**

**DEMO XÂY DỰNG NFA-ε VÀ KIỂM TRA MỘT CHUỖI CÓ THUỘC NFA-ε ĐÃ CHO KHÔNG?**

**NHÓM BÁO CÁO 22**

|  |  |  |
| --- | --- | --- |
| **SỐ THỨ TỰ TRONG DANH SÁCH** | **MSSV** | **HỌ TÊN** |
| **22** | **B2113316** | **Phạm Hoàng Phong** |
| **28** | **B2113342** | **Lê Sỹ Anh Tấn** |
| **31** | **B2113394** | **Nguyễn Minh Tâm** |

**HK 2, NH 2023 2024**

Cần Thơ, 04/2024

**NHẬN XÉT CỦA GIẢNG VIÊN**

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**LỜI CẢM ƠN**

Đầu tiên, nhóm chúng em xin gửi lời cám ơn chân thành đến Trường Đại Học Cần Thơ, cám ơn Trường CNTT & TT, cám ơn Khoa KHMT đã tạo nên một môi trường học tập chất lượng, không những trang bị cho sinh viên kiến thức chuyên môn vững chắc mà các thầy cô lúc nào cũng luôn dành một sự quan tâm đến sinh viên. Đặc biệt, nhóm chúng em xin gửi lời cám ơn sâu sắc đến giáo viên hướng dẫn – Cô Phạm Xuân Hiền đã truyền đạt những kiến thức và kinh nghiệm cho nhóm chúng em trong suốt quá trình thực hiện bài báo cáo vừa qua. Trong thời gian thực hiện đề tài của mình với sự hướng dẫn của Cô, nhóm chúng em đã tích luỹ và trau dồi cho mình nhiều kĩ năng hơn. Điều đó sẽ trở thành một bước đà vững chắc cho sự phát triển của nhóm chúng em sau này.

Tin học lý thuyết là một môn vô cùng bổ ích và có tính thực tế cao. Ngoài việc củng cố kiến thức cơ sở, kiến thức chuyên ngành cho sinh viên Trường Công nghệ thông tin & Truyền thông nói chung và cho sinh viên ngành Khoa học máy tính nói riêng, môn học này còn giúp nhóm chúng em cải thiện khả năng lập trình và tư duy sáng tạo. Ngoài ra, còn áp dụng được những kiến thức đã học tập từ các học phần trong Chương trình đào tạo vào đề tài để tạo nên một sản phẩm hoàn thiện.

Do nhóm chúng em chưa có nhiều kinh nghiệm, khả năng hiểu biết vẫn còn hạn hẹp nên trong quá trình thực hiện đề tài chắc chắn sẽ không tránh khỏi những thiếu sót. Nhờ sự giúp đỡ nhiệt tình từ Cô, nhóm chúng em đã hoàn thành được đề tài của mình. Vì vậy, rất mong nhận được sự nhận xét, ý kiến đóng góp từ Cô để bài Báo cáo của nhóm chúng em được hoàn thiện hơn.

Nhóm chúng em xin chân thành cám ơn.

Cần Thơ, ngày 21 tháng 04 năm 2024

Người viết

Phạm Hoàng Phong

Lê Sỹ Anh Tấn

Nguyễn Minh Tâm

**MỤC LỤC**

[PHẦN GIỚI THIỆU 7](#_Toc164670436)

[1. Đặt vấn đề 7](#_Toc164670437)

[2. Lịch sử giải quyết vấn đề 7](#_Toc164670438)

[3. Mục tiêu đề tài 8](#_Toc164670439)

[4. Đối tượng và phạm vi nghiên cứu 8](#_Toc164670440)

[5. Phương pháp nghiên cứu 8](#_Toc164670441)

[6. Kết quả đạt được 9](#_Toc164670442)

[7. Bố cục luận văn 9](#_Toc164670443)

[PHẦN NỘI DUNG 10](#_Toc164670444)

[CHƯƠNG 1 10](#_Toc164670445)

[MÔ TẢ BÀI TOÁN 10](#_Toc164670446)

[1. Mô tả chi tiết bài toán 10](#_Toc164670447)

[2. Vấn đề liên quan đến bài toán Xây dựng NFA-ε 11](#_Toc164670448)

[2.1. Độ phức tạp của quy trình xây dựng 11](#_Toc164670449)

[2.2. Xử lý dịch chuyển ε 11](#_Toc164670450)

[2.3. Kiểm tra tính hợp lệ 11](#_Toc164670451)

[2.4. Độ dài đường đi 11](#_Toc164670452)

[2.5. Tối ưu hoá 11](#_Toc164670453)

[2.6. Tính tổng quát của thuật toán xây dựng NFA-ε 11](#_Toc164670454)

[3. Giải pháp cho bài toán 11](#_Toc164670455)

[4. Công nghệ sử dụng 12](#_Toc164670456)

[4.1. Ngôn ngữ lập trình Python 12](#_Toc164670457)

[4.2. Thư Viện Tkinter 12](#_Toc164670458)

[4.3. Visual Studio Code (VSCode) 13](#_Toc164670459)

[CHƯƠNG 2 14](#_Toc164670460)

[THIẾT KẾ VÀ CÀI ĐẶT 14](#_Toc164670461)

[1. Thiết kế hệ thống 14](#_Toc164670462)

[1.1. NFAe.py 15](#_Toc164670463)

[1.2. Handle.py 15](#_Toc164670464)

[1.3. GUI.py 15](#_Toc164670465)

[1.4. nfa.txt 16](#_Toc164670466)

[2. Thiết kế và cài đặt giải thuật 17](#_Toc164670467)

[3. Giao diện hệ thống 21](#_Toc164670468)

[CHƯƠNG 3 24](#_Toc164670469)

[KIỂM THỬ VÀ ĐÁNH GIÁ 24](#_Toc164670470)

[1. Mục tiêu 24](#_Toc164670471)

[2. Nghi thức kiểm tra 24](#_Toc164670472)

[a. Kiểm thử tính năng nhập dữ liệu đầu vào từ file: 24](#_Toc164670473)

[b. Kiểm thử tính năng kiểm tra chuỗi nhập vào: 25](#_Toc164670474)

[PHẦN KẾT LUẬN 27](#_Toc164670475)

[1. Kết quả đạt được 27](#_Toc164670476)

[2. Hướng phát triển 27](#_Toc164670477)

[TÀI LIỆU THAM KHẢO 28](#_Toc164670478)

**DANH MỤC HÌNH**

[Hình 1. NFA với ε dịch chuyển 10](#_Toc164637071)

[Hình 2. Cấu trúc thư mục nguồn của chương trình 14](#_Toc164637072)

[Hình 3. Quy ước cấu trúc file text chứa thông tin của NFA-ε 16](#_Toc164637073)

[Hình 4. Tập tin văn bản đầu vào đúng cấu trúc được quy ước 16](#_Toc164637074)

[Hình 5. Phương thức GetFile 17](#_Toc164637075)

[Hình 6. Phương thức CheckString 18](#_Toc164637076)

[Hình 7. Phương thức ShowImage 18](#_Toc164637077)

[Hình 8. Phương thức MoveE 19](#_Toc164637078)

[Hình 9. Phuơng thức Move 20](#_Toc164637079)

[Hình 10. Phương thức MoveWithString 20](#_Toc164637080)

[Hình 11. Phương thức IsAccept 21](#_Toc164637081)

[Hình 12. Giao diện khi chạy chương trình 21](#_Toc164637082)

[Hình 13. Giao diện khi người chọn tập tin đầu vào 22](#_Toc164637083)

[Hình 14. Giao diện khi người dùng đã chọn xong tập tin đầu vào 22](#_Toc164637084)

[Hình 15. Giao diện kết quả kiểm tra chuỗi 23](#_Toc164637085)

**DANH MỤC BẢNG**

[Bảng 1: Kết nhập tính năng nhập dữ liệu đầu vào từ file 24](#_Toc164592570)

[Bảng 2: Kết xuất tính năng nhập dữ liệu đầu vào từ file 25](#_Toc164592571)

[Bảng 3: Kết nhập tính năng kiểm tra chuỗi nhập vào 25](#_Toc164592572)

[Bảng 4: Kết xuất tính năng kiểm tra chuỗi nhập vào 26](#_Toc164592573)

**ABSTRACT**

The demonstration on constructing and testing strings on NFA-ε aims to provide an overview of the model's operation and its practical applications in formal language and automation. Throughout the demo, participants will be guided from basic concepts to string testing algorithms. They will engage in constructing NFA-ε and applying string testing algorithms to gain a deeper understanding of their functionality and practical applications. Additionally, the demo focuses on bridging theory to practice, enabling participants to develop practical skills in creating and testing NFA-ε. Moreover, presenting specific examples and applications in areas such as text analysis will spark interest and foster a profound understanding of the potential of NFA-ε in real-world scenarios. Ultimately, through this demo, participants will not only grasp knowledge about NFA-ε but also comprehend how to apply and leverage this model to address real-world challenges in automation and formal language.

**TÓM TẮT**

Demo về xây dựng và kiểm tra chuỗi trên NFA-ε nhằm cung cấp cái nhìn tổng quan về hoạt động của mô hình và ứng dụng thực tế của nó trong lĩnh vực ngôn ngữ hình thức và tự động hóa. Trong quá trình demo, người tham gia sẽ được hướng dẫn từ các khái niệm cơ bản đến các thuật toán kiểm tra chuỗi. Họ sẽ thực hành xây dựng NFA-ε và áp dụng thuật toán kiểm tra chuỗi để hiểu rõ hơn về hoạt động của thuật toán và ứng dụng thực tiễn của nó. Bên cạnh đó, demo cũng tập trung vào việc áp dụng kiến thức từ lý thuyết sang thực hành, giúp người tham gia xây dựng kỹ năng thực tế trong việc tạo ra và kiểm tra NFA-ε. Đồng thời, việc trình bày các ví dụ và ứng dụng cụ thể trong các lĩnh vực như phân tích văn bản sẽ khơi gợi sự quan tâm và nhận thức sâu sắc về tiềm năng của NFA-ε trong thực tế. Cuối cùng, thông qua quá trình demo này, người tham gia sẽ không chỉ nắm vững kiến thức về NFA-ε mà còn hiểu rõ hơn về cách áp dụng và tận dụng mô hình này để giải quyết các vấn đề thực tế trong lĩnh vực tự động hóa và ngôn ngữ hình thức.

# PHẦN GIỚI THIỆU

## 1. Đặt vấn đề

Trong thời đại công nghệ ngày nay, vấn đề về xử lý chuỗi là một thách thức quan trọng trong nhiều lĩnh vực của khoa học máy tính và công nghệ thông tin. Với sự phổ biến của dữ liệu dưới dạng chuỗi trong các ứng dụng web, ứng dụng di động, trí tuệ nhân tạo và nhiều lĩnh vực khác, việc hiểu và xử lý chuỗi trở thành một yếu tố quyết định đối với sự thành công của các dự án công nghệ.

Trong xử lý chuỗi, một thách thức phổ biến là làm thế nào để biểu diễn và xử lý các mẫu phức tạp một cách hiệu quả. Non-deterministic Finite Automata với epsilon-transitions (NFA-ε) trở thành một công cụ quan trọng trong việc giải quyết vấn đề này. Bằng cách cho phép biểu diễn các mẫu chuỗi có cấu trúc phức tạp và xử lý các trạng thái không xác định, NFA-ε mở ra những khả năng mới trong việc xử lý chuỗi. Tính linh hoạt và tiết kiệm bộ nhớ của NFA-ε làm cho nó trở thành lựa chọn lý tưởng cho nhiều ứng dụng xử lý chuỗi, từ xử lý ngôn ngữ tự nhiên đến phân tích dữ liệu. Sự kết hợp giữa vấn đề xử lý chuỗi và sức mạnh của NFA-ε tạo nên một phương tiện mạnh mẽ để giải quyết các thách thức trong lĩnh vực này và tạo ra những giải pháp hiệu quả.

Vấn đề của demo xây dựng NFA-ε và kiểm tra một chuỗi có thuộc NFA-ε đã cho hay không là cung cấp một phương pháp hiệu quả để xây dựng và kiểm tra chuỗi trong một NFA-ε (Nondeterministic Finite Automaton with ε-transitions).

NFA-ε là một mô hình tự động hóa quan trọng trong lĩnh vực ngôn ngữ hình thức và tự động hóa. Tuy nhiên, việc xây dựng và kiểm tra chuỗi trong một NFA-ε có thể trở nên phức tạp và đòi hỏi nhiều công sức.

Bằng cách giải quyết vấn đề này, chúng ta có thể cung cấp một công cụ mạnh mẽ và linh hoạt cho những người làm việc trong lĩnh vực tự động hóa và ngôn ngữ hình thức, giúp họ dễ dàng xây dựng và kiểm tra các mô hình NFA-ε một cách hiệu quả. Điều này cũng có thể giúp tăng cường hiệu suất và sự chính xác trong các ứng dụng thực tế như phân tích văn bản, xử lý ngôn ngữ tự nhiên và biên dịch ngôn ngữ.

## 2. Lịch sử giải quyết vấn đề

Trong quá trình phát triển của lĩnh vực tự động hóa và ngôn ngữ hình thức, việc xây dựng và kiểm tra chuỗi trong NFA-ε (Nondeterministic Finite Automaton with ε-transitions) đã trở thành một vấn đề quan trọng và được quan tâm. Ban đầu, việc thực hiện này thường phải dựa vào phương pháp thủ công hoặc sử dụng các công cụ cơ bản. Tuy nhiên, với sự phát triển của công nghệ và nhu cầu ngày càng tăng về tự động hóa quy trình này, các nhà nghiên cứu và nhà phát triển đã bắt đầu tìm kiếm các phương pháp và công nghệ mới để giải quyết vấn đề này.

Sự xuất hiện của ngôn ngữ lập trình và các thư viện hỗ trợ, cùng với sự phát triển của các thuật toán và kỹ thuật trong lĩnh vực lập trình và tự động hóa, đã tạo ra các công cụ mạnh mẽ và hiệu quả cho việc xây dựng và kiểm tra NFA-ε. Ngày nay, việc xây dựng và kiểm tra chuỗi trong NFA-ε đã trở nên phổ biến và dễ dàng hơn nhờ vào sự tiến bộ trong công nghệ và sự phát triển của các công cụ và tài liệu hướng dẫn. Các nhà phát triển có thể sử dụng các thư viện và framework có sẵn hoặc viết mã từ đầu để thực hiện quy trình này một cách hiệu quả và linh hoạt.

## 3. Mục tiêu đề tài

Đề tài “Demo xây dựng NFA-ε và kiểm tra một chuỗi có thuộc NFA-ε hay không?” với mục tiêu là xây dựng được một chương trình với giao diện đơn giản và bao gồm những chức năng sau:

* Đọc file văn bản chứa thông tin của một NFA-ε.
* Xây dựng NFA-ε từ thông tin của file văn bản.
* Hiển thị hình ảnh mô tả NFA-ε.
* Kiểm tra một chuỗi đầu vào do người dùng nhập.

## 4. Đối tượng và phạm vi nghiên cứu

Trong phạm vi đề tài báo cáo này, sau khi nghiên cứu cơ sở lý thuyết và các đề tài liên quan, nhóm báo cáo đã xác định được đối tượng và phạm vi nghiên cứu sẽ bao gồm:

* Nghiên cứu cách xây dựng một chương trình ứng dụng với ngôn ngữ Python.
* Tìm hiểu cách xây dựng giao diện của chương trình với thư viện tkinter và cách để biếu diễn Automata với thư viện Automathon.
* Nghiên cứu về lý thuyết Automata, đặc biệt là về NFA-ε.

## 5. Phương pháp nghiên cứu

Đề tài này tổng hợp các kiến thức đã học về lý thuyết Automata và ngôn ngữ hình thức, kết hợp với ngôn ngữ lập trình Python cùng với các thư viện hỗ trợ để xây dựng nên một chương trình hoạt động đúng với yêu cầu. Vì thế sẽ có những phương pháp nghiên cứu như sau:

* Tìm hiểu về lý thuyết Automata và ngôn ngữ hình thức thông qua giáo trình, slide bài giảng, bài tập học được trên lớp.
* Tìm hiểu cách xây dựng giao diện cho một chương trình bằng thư viện tkinter thông qua tài liệu chính thức trên trang chủ.
* Tìm hiểu cách xây dựng một chương trình ứng dụng có thể áp dụng các chức năng cho phép người dùng có thể mở file, đọc file và nhập chuỗi đầu vào.
* Thiết kế được chức năng hiển thị hình ảnh mô phỏng một NFA-ε.

## 6. Kết quả đạt được

* **Hiểu biết sâu sắc về NFA-ε:** Chúng em có khả năng hiểu rõ hơn về cấu trúc và hoạt động của NFA-ε, bao gồm các trạng thái, chuyển đổi và epsilon-chuyển đổi.
* **Năng lực xây dựng NFA-ε:** Chúng em có thể xây dựng một NFA-ε từ các biểu thức chính quy hoặc bảng chuyển đổi. Họ có khả năng thêm và chỉnh sửa các trạng thái và chuyển đổi để tạo ra một mô hình NFA-ε hoàn chỉnh.
* **Kiểm tra chuỗi trong NFA-ε:** Chúng em có khả năng áp dụng thuật toán để kiểm tra xem một chuỗi có thuộc NFA-ε đã cho hay không. Họ có thể thực hiện quy trình này một cách chính xác và hiệu quả.
* **Áp dụng trong thực tế:** Những kiến thức và kỹ năng họ học được từ demo có thể được áp dụng vào các ứng dụng thực tế như phân tích văn bản, xử lý ngôn ngữ tự nhiên và kiểm tra cú pháp của ngôn ngữ lập trình.
* **Tăng cường sự tự tin:** Demo cung cấp một cơ hội cho chúng em để thực hành và làm quen với việc làm việc với NFA-ε. Điều này có thể giúp chúng em cảm thấy tự tin hơn khi tiếp cận với các vấn đề liên quan trong lĩnh vực tự động hóa và ngôn ngữ hình thức.

## 7. Bố cục luận văn

**Phần giới thiệu**

Giới thiệu tổng quát về đề tài.

**Phần nội dung**

**Chương 1**: Mô tả bài toán.

**Chương 2**: Thiết kế, cài đặt giải thuật, biễu diễn cơ sở dữ liệu, trình bày các bước xây dựng hệ thống bằng phương pháp lọc cộng tác.

**Chương 3**: Kiểm thử hệ thống và đánh giá độ chính xác, tốc độ của hệ thống.

**Phần kết luận**

Trình bày kết quả đạt được và hướng phát triển hệ thống.

# PHẦN NỘI DUNG

# CHƯƠNG 1

# MÔ TẢ BÀI TOÁN

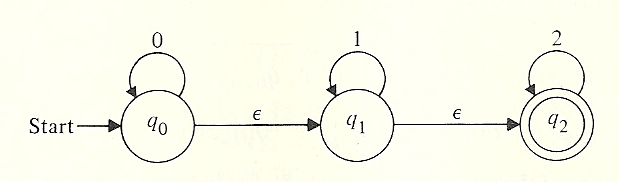
## 1. Mô tả chi tiết bài toán

Bài toán xây dựng NFA-ε (NFA với ε dịch chuyển) là một dạng bài toán quan trọng trong lý thuyết Automata và ngôn ngữ hình thức. NFA-ε là một dạng của Automata hữu hạn không đơn định (NFA) mà ở đó có thể di chuyển đến nhiều trạng thái khác nhau từ một trạng thái và có thể di chuyển trên bước ε, tức là ký tự rỗng. Việc này làm cho việc xác định xem một chuỗi có thuộc NFA-ε hay không trở nên phức tạp hơn.

Một **NFA-ε** gồm các thành phần:

* **States (Trạng thái):** Một tập hữu hạn các trạng thái, ký hiệu Q.
* **Alphabet (Bảng chữ cái):** Một tập hữu hạn các ký tự, ký hiệu Σ.
* **Transition Function (Hàm chuyển trạng thái):** Hàm chuyển trạng thái, thường được ký hiệu là δ. Định nghĩa cách Automata chuyển từ trạng thái này sang trạng thái khác dựa tren ký tự từ bảng chữ cái Alphabet hoặc bước ε.
* **Start State (Trạng thái bắt đầu):** Một trạng thái khởi đầu, từ đây Automata bắt đầu xử lý chuỗi được đưa vào, thường ký hiệu là q0.
* **Accept States (Trạng thái kết thúc):** Một tập các trạng thái mà nếu Automata kết thúc ở một trong số các trạng thái này thì chuỗi đầu vào sẽ được chấp nhận, ký hiệu F.

Để sinh ra một **NFA-ε**, chúng ta cần xác định đầy đủ các thành phần trên và có thể biểu diễn bằng đồ thị với các trạng thái là các đỉnh của đồ thị và cung có hướng biểu diễn các bước chuyển của **NFA-ε**.



Hình 1. NFA với ε dịch chuyển

Bên cạnh đó, kiểm tra một chuỗi đầu vào có được chấp nhận bởi NFA-ε đã cho hay không cũng là một dạng bài toán quan trọng trong lĩnh vực ngôn ngữ hình thức. Quy trình để kiểm tra như sau:

* **Khởi tạo:** bắt đầu từ Trạng thái bắt đầu (Start state) của NFA-ε.
* **Duyệt chuỗi:** duyệt tuần tự qua từng ký tự của chuỗi.
* **Mở rộng trạng thái:** với mỗi ký tự trong chuỗi và mỗi trạng thái hiện tại của NFA-ε, xem xét tất cả các trạng thái có thể dịch chuyển được.
* **Kiểm tra trạng thái kết thúc:** sau khi duyệt toàn bộ chuỗi, kiểm tra xem có trạng thái nào nằm trong tập trạng thái kết thúc hay không và đưa ra kết luận.

## 2. Vấn đề liên quan đến bài toán Xây dựng NFA-ε

### 2.1. Độ phức tạp của quy trình xây dựng

Xây dựng NFA-ε từ một mô tả ngôn ngữ hoặc từ một biểu thức chính quy có thể phức tạp, đặc biệt là khi ngôn ngữ hoặc biểu thức có cấu trúc phức tạp. Thuật toán cần phải được thiết kế để xử lý các trường hợp đặc biệt và đảm bảo tính chính xác và hiệu quả.

### 2.2. Xử lý dịch chuyển ε

Việc xử lý chuyển tiếp ε có thể phức tạp, đặc biệt là trong quá trình xác định các trạng thái đến và từ đó. Đôi khi, sự tồn tại của các chuyển tiếp ε có thể làm cho việc phân tích NFA-ε trở nên khó khăn.

### 2.3. Kiểm tra tính hợp lệ

Khi xây dựng NFA-ε từ một mô tả ngôn ngữ hoặc từ một biểu thức chính quy, điều quan trọng là để đảm bảo rằng NFA-ε được sinh ra là hợp lệ, tức là nó có thể nhận diện chính xác tất cả các chuỗi thuộc ngôn ngữ mục tiêu.

### 2.4. Độ dài đường đi

Một vấn đề khác là quản lý độ dài của đường đi trong NFA-ε. Đặc biệt là trong các trường hợp có thể xảy ra lặp lại không cần thiết hoặc đường đi dài và phức tạp.

### 2.5. Tối ưu hoá

Một vấn đề quan trọng khác liên quan đến NFA-ε là tối ưu hóa NFA-ε, tức là loại bỏ các trạng thái không cần thiết hoặc làm cho NFA-ε có kích thước nhỏ hơn nhưng vẫn biểu diễn chính xác cùng một ngôn ngữ.

### 2.6. Tính tổng quát của thuật toán xây dựng NFA-ε

Thuật toán xây dựng NFA-ε cần phải làm việc với các ngôn ngữ và biểu thức chính quy đa dạng. Điều này yêu cầu thuật toán phải có tính tổng quát và có khả năng xử lý nhiều loại ngôn ngữ và biểu thức.

## 3. Giải pháp cho bài toán

* **Dùng thuật toán hiệu quả:** Phát triển và triển khai thuật toán hiệu quả để xây dựng NFA-ε từ một mô tả ngôn ngữ hoặc từ một biểu thức chính quy. Thuật toán này nên có độ phức tạp thấp và có khả năng xử lý các trường hợp đặc biệt một cách hiệu quả.
* **Tối ưu hóa sau quá trình xây dựng:** Áp dụng các kỹ thuật tối ưu hóa sau khi xây dựng NFA-ε để loại bỏ các trạng thái và các bước chuyển không cần thiết, giảm thiểu độ phức tạp của NFA-ε và tiết kiệm tài nguyên.
* **Kiểm tra tính hợp lệ:** Xây dựng các công cụ và phương pháp kiểm tra tính hợp lệ của NFA-ε, đảm bảo rằng nó có thể nhận diện chính xác tất cả các chuỗi thuộc ngôn ngữ mục tiêu một cách đáng tin cậy.
* **Sử dụng kỹ thuật tăng cường:** Áp dụng kỹ thuật tăng cường để cải thiện hiệu suất và tính tổng quát của thuật toán xây dựng NFA-ε, cho phép nó xử lý nhiều loại ngôn ngữ và biểu thức một cách hiệu quả.
* **Tối ưu hóa hiệu suất và sử dụng tài nguyên:** Tìm cách để tối ưu hóa hiệu suất và sử dụng tài nguyên của NFA-ε, bao gồm việc giảm bớt bộ nhớ được sử dụng và tối ưu hóa thời gian tính toán.

## 4. Công nghệ sử dụng

### 4.1. Ngôn ngữ lập trình Python

Python là ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991. Python được thiết kế với ưu điểm là dễ đọc, dễ học và dễ nhớ, có ngôn ngữ hình thức sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình và được dùng rộng rãi trong phát triển trí tuệ nhân tạo.

Python hỗ trợ nhiều phong cách lập trình khác nhau, từ hướng đối tượng (OOP) đến lập trình hàm hay lập trình thủ tục và từ đó cho phép lập trình viên có thể phát triển nhiều loại ứng dụng khác nhau.

Python cũng hỗ trợ rất nhiều thư viện và framework phong phú, mạnh mẽ cũng như cộng đồng người dùng to lớn giúp cho quá trình lập trình của chúng ta trở nên dễ dàng và nhanh chóng hơn. Trong phạm vi đề tài này, chúng ta sẽ sử dụng 2 thư viện của python gồm tkinter dùng để tạo giao diện và Automathon dùng để biểu diễn NFA-ε dễ dàng hơn.

### 4.2. Thư Viện Tkinter

Thư viện tkinter là một trong những thư viện phổ biến nhất để phát triển giao diện người dùng đồ họa (GUI) trong Python. Tkinter cung cấp các công cụ cần thiết để tạo ra các ứng dụng có giao diện đồ họa trực quan và dễ sử dụng.

Dù là một thư viện nhỏ gọn, tkinter vẫn cung cấp đầy đủ các thành phần cần thiết để xây dựng giao diện, bao gồm các widget như button, label, entry, và canvas. Bạn cũng có thể tùy chỉnh giao diện bằng cách sử dụng các thuộc tính và phương thức có sẵn trong các widget này.

Với sự linh hoạt và tính tiện dụng, tkinter đã trở thành lựa chọn phổ biến cho việc phát triển các ứng dụng desktop trong Python, từ các ứng dụng nhỏ đến các dự án lớn hơn.

### 4.3. Visual Studio Code (VSCode)

VSCode là một trình soạn thảo mã nguồn được phát triển bởi Microsoft dành cho Windows, Linux và macOS. Nó hỗ trợ chức năng debug, đi kèm với Git, có chức năng nổi bật cú pháp (syntax highlighting), tự hoàn thành mã thông minh, snippets, và cải tiến mã nguồn, VSCode có thể hỗ trợ hầu hết các ngôn ngữ lập trình hiện nay. Với khả năng tuỳ biến mạnh mẽ cũng như có một kho tiện ích mở rộng đồ sộ sẽ hỗ trợ và giúp ích cho việc lập trình của người dùng 1 cách tối đa. VSCode cũng có một cộng đồng người dùng lớn mạnh sẵn sàn hỗ trợ với bất kỳ vấn đề nào mà chúng ta gặp phải.

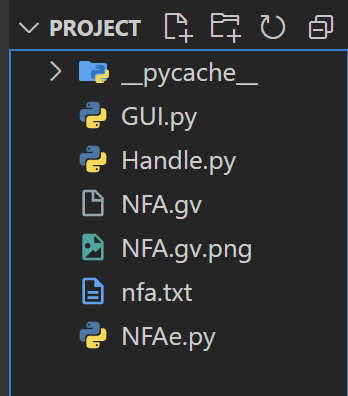
# CHƯƠNG 2

# THIẾT KẾ VÀ CÀI ĐẶT

## 1. Thiết kế hệ thống

Chương trình “Demo xây dựng NFA-ε và kiểm tra một chuỗi có thuộc NFA-ε đã cho hay không” là một chương trình đơn giản với giao diện được xây dựng bằng Python sử dụng thư viện tkinter và một số thư viện khác như automathon để biểu diễn một NFA-ε bằng đồ thị đồng thời sử dụng class NFAe để biểu diễn và dịch chuyển các trạng thái của một NFA-ε. Chương trình bao gồm các chức năng:

* **Xây dựng NFA-ε hoàn chỉnh từ tập tin văn bản đầu vào:** Người dùng sẽ chọn một file văn bản đầu vào (.txt) có lưu trữ các thông tin của NFA-ε theo cấu trúc đã được quy ước, hệ thống sẽ tự động nhận diện và xây dựng NFA-ε bằng thuật toán đã được thiết lập sẵn.
* **Biểu diễn NFA-ε:** Sau khi đã xây dựng được NFA-ε từ tập tin đầu vào, hệ thống sẽ hiển thị hình ảnh mô tả NFA-ε đó ở dạng đồ thị với các trạng thái là các đỉnh và các bước dịch chuyển là các cung có hướng.
* **Kiểm tra một chuỗi đầu vào:** Người dùng có thể nhập vào một chuỗi để hệ thống kiểm tra xem chuỗi đó có được chấp nhận bởi NFA-ε vừa được xây dựng hay không. Hệ thống sẽ cho ra kết quả và hiển thị tập trạng thái cuối cùng của chuỗi.



Hình 2. Cấu trúc thư mục nguồn của chương trình

### 1.1. NFAe.py

Tập tin này bao gồm Class NFAe dùng để định nghĩa một NFA-ε, bao gồm các thuộc tính và phương thức sau:

* **Các thuộc tính:** 
  + **Var:** tương ứng với Tập các trạng thái của NFA-ε.
  + **Sigma:** tương ứng với Bảng chữ cái.
  + **MoveMap:** tương ứng với Hàm chuyển trạng thái.
  + **Start:** tương ứng với trạng thái bắt đầu.
  + **End:** tương ứng với Tập trạng thái kết thúc.
* **Các phương thức:**
  + ***MoveE(self):*** phương thức được dùng khi NFA-ε dịch chuyển trên ε.
  + ***Move(self, input):*** phương thức được dùng khi NFA-ε dịch chuyển trên nhãn đầu vào.
  + ***MoveWithString(self, string):*** phương thức dùng để kiểm tra một chuỗi ký tự bằng việc gọi ***Move(self, input)*** nhiều lần.
  + ***isAccept(self, input):*** phương thức được dùng để kiểm tra một chuỗi đầu vào có được đón nhận bởi NFA-ε không, trả về True khi chuỗi input được đón nhận, ngược lại trả về False.
  + ***resetCurrent(self):*** phương thức dùng để chuyển NFA-ε quay về trạng thái bắt đầu.

### 1.2. Handle.py

Tập tin này bao gồm các phương thức xử lý sau:

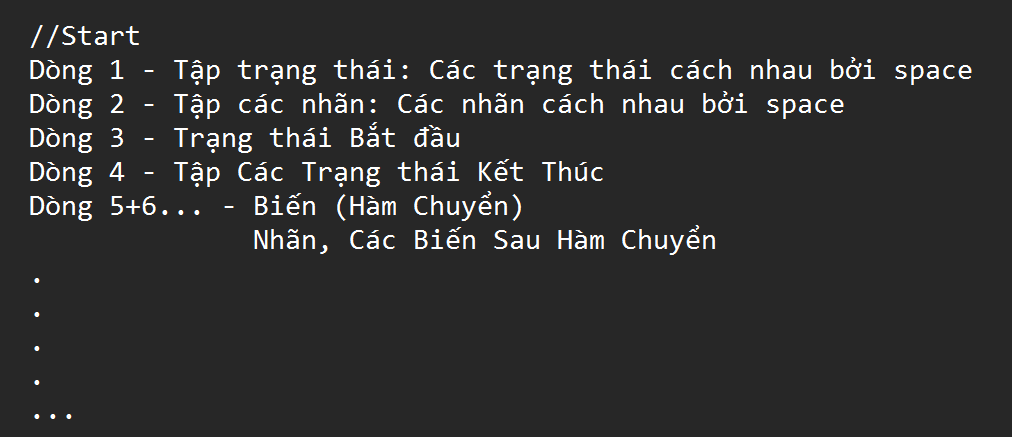
* ***CheckString(event = None, entry: Entry = None):*** phương thức được dùng để kiểm tra chuỗi đầu vào do người dùng nhập vào. Sau khi kiểm tra sẽ hiển thị kết quả cùng với tập trạng thái kết thúc.
* ***GetFile(window):*** phương thức được dùng để đọc file text chứa các thông tin về NFA-ε có cấu trúc đã được quy ước sẵn và tiến hành xây dựng NFA-ε sử dụng Class NFAe.
* ***ShowImage(window):*** phương thức dùng để hiển thị hình ảnh mô tả NFA-ε sau khi được xây dựng.

### 1.3. GUI.py

Tập tin này chỉ bao gồm các dòng lệnh dùng để tạo ra 1 giao diện đơn giản sử dụng thư viện tkinter để thực thi chương trình Xây dựng NFA-ε và kiểm tra chuỗi sử dụng Class NFAe và các phương thức đã được khai báo ở 2 tập tin Handle.py và NFAe.py.

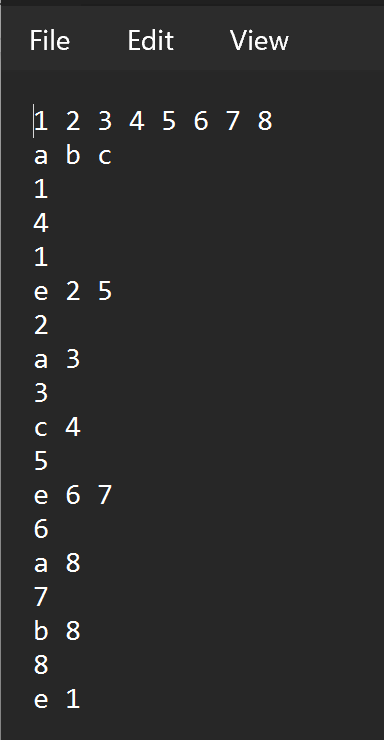
### 1.4. nfa.txt

Tập tin văn bản chứa các thông để xây dựng nên một NFA-ε hoàn chỉnh. Tập tin này có cấu trúc đã được quy ước sẵn để quá trình đọc file trở nên dễ dàng và chính xác hơn.



Hình 3. Quy ước cấu trúc file text chứa thông tin của NFA-ε

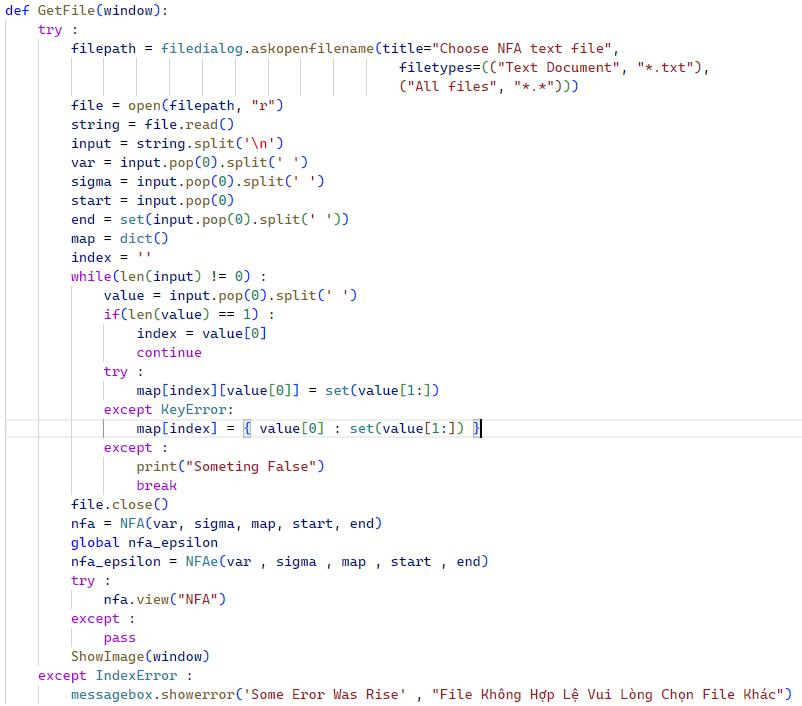
Sau đây là ví dụ của một tập tin văn bản đầu vào đúng theo cấu trúc trên:



Hình 4. Tập tin văn bản đầu vào đúng cấu trúc được quy ước

## 2. Thiết kế và cài đặt giải thuật

**2.1. Xây dựng NFA-ε từ file txt:**



Hình 5. Phương thức GetFile

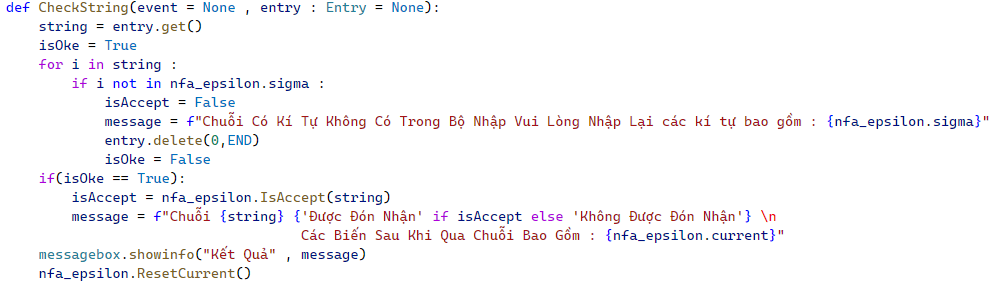
Để xây dựng một NFA-ε, chúng ta cần có 5 thành phần đã được nêu ở phần mô tả bài toán, những thông tin về 5 thành phần này sẽ được lưu theo cấu trúc đã được quy ước trong một file văn bản có tên là “nfa.txt”. Phương thức ***GetFile(window)*** sẽ chịu trách nhiệm cho việc đọc file này và lưu những thông đó để tiến hành xây dựng NFA-ε.

Khi phương thức được gọi qua một nút bấm trên giao diện, một hộp thoại csẽ mở ra và yêu cầu người dùng chọn tập tin đúng cấu trúc. Sau khi chọn, phương thức sẽ đọc toàn bộ nội dung trong tập tin đó và tiến hành tách theo từng dòng bằng phương thức split() của chuỗi với tham số là ký tự “\n”, kết quả thu được là một danh sách mà mỗi phần tử là một dòng được tách ra từ tập tin văn bản trên.

Sau đó sẽ lần lượt đọc và lưu giá trị của từng phần tử trong danh đó vào 5 biến thành phần của một NFA-ε, sau khi lưu toàn bộ giá trị, nó sẽ tiến hành xây dựng NFA-ε với Class NFAe đã được định nghĩa và lưu với tên biến “nfa\_epsilon”.

Ở đây cũng khởi tạo kèm theo một biến “nfa” sử dụng lớp NFA có sẵn của thư viện Automathon để tạo và xuất ra một tập tin hình ảnh mô tả NFA-ε để hiển thị lên giao diện người dùng.

**2.2. Kiểm tra chuỗi nhập vào:**



Hình 6. Phương thức CheckString

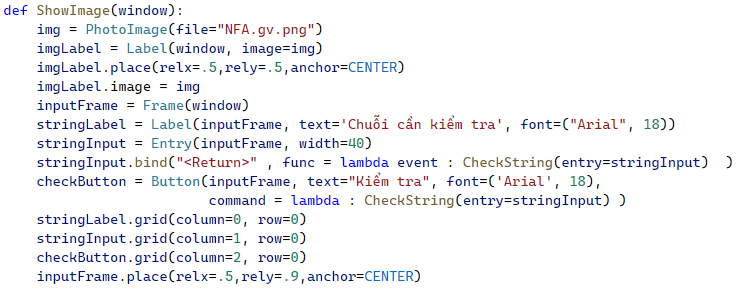
Phương thức ***CheckString(event = None, entry : Entry = None)*** khi được gọi sẽ lấy chuỗi đầu vào mà người dùng đã nhập.

Ta sẽ có biến boolean “**isOke** **= True**” để xác định chuỗi hợp lệ, kiểm tra lần lượt từng ký tự trong chuỗi, nếu phát hiện ký tự không hợp lệ (không nằm trong Σ) thì sẽ gán giá trị False cho isOke.

Ta cũng sẽ có biến boolean “**isAccept**” để kiểm tra chuỗi có được đón nhận không và chỉ xét khi chuỗi hợp lệ (isOke = True).

Cuối cùng phương thức sẽ trả về kết quả là một Message Box thông báo kết quả của chuỗi được kiểm tra và reset trạng thái của NFA-ε.

**2.3. Hiển thị hình ảnh:**

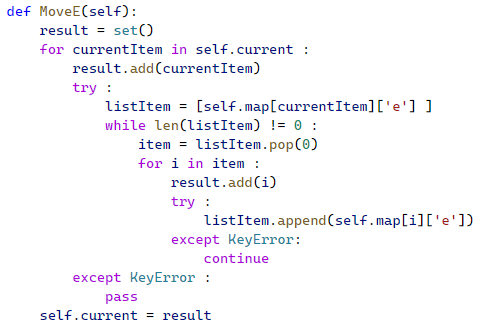


Hình 7. Phương thức ShowImage

Sau khi xây dựng được NFA-ε, chương trình cũng sẽ xuất ra được 1 tập tin hình ảnh mô tả NFA-ε với tên là “NFA.gv.png”. Phương thức ShowImage(window) sẽ chịu trách nhiệm cho việc hiển thị hình ảnh mô tả này trên giao diện ngay sau người dùng đã chọn tập tin đầu vào thành công.

**2.4. Các phương thức trong lớp NFAe:**

**2.4.1. Bước dịch chuyển trên ε:**

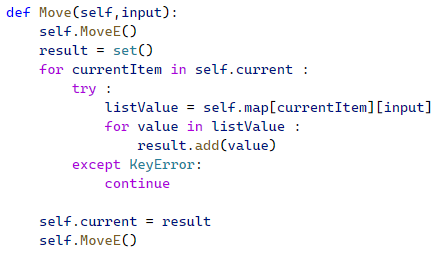


Hình 8. Phương thức MoveE

Đầu tiên, khởi tạo một tập hợp “result” lưu trữ kết quả. Duyệt qua tập các trạng thái hiện tại đang có (self.current). Với mỗi trạng thái, thêm vào tập result. Sau đó lấy các hàm chuyển với ε của trạng thái đó, ta được “listItem” là tập các trạng thái sau khi dịch chuyển trên ε. Với mỗi phần tử ứng với một trạng thái trong listItem, ta thêm vào tập result và kiểm tra xem còn tiếp tục dịch chuyển trên ε không, nếu còn thì tiếp tục thêm vào listItem để xử lý sau.

Cuối cùng điều chỉnh tập trạng thái hiện tại thành tập result.

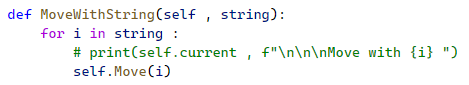
**2.4.2. Bước dịch chuyển trên 1 nhãn đầu vào:**



Hình 9. Phuơng thức Move

Đầu tiên, dịch chuyển đầu vào trên ε. Sau đó, dịch chuyển các trạng thái dựa vào nhãn input và hàm chuyển trạng thái. Sau cùng, dịch chuyển trên ε một lần nữa là hoàn thành bước chuyển với 1 nhãn.

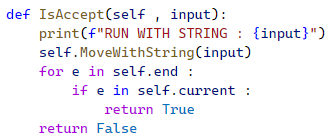
**2.4.3. Bước dịch chuyển trên chuỗi đầu vào:**



Hình 10. Phương thức MoveWithString

Phương thức MoveWithString(self, string) nhận input là một chuỗi đầu vào do người dùng nhập vào và tiến hành chạy vòng lặp để dịch chuyển trên từng ký tự của chuỗi đầu vào sử dụng phương thức Move(self, input) ở trên.

**2.4.4. Kiểm Tra Một Chuỗi Có Được Đón Nhận:**



Hình 11. Phương thức IsAccept

Hàm Kiểm tra các trạng thái hiện tại có nằm trong tập kết thúc. Nếu có trả về kết quả là True, còn nếu không có trong tập kết thúc thì trả về kết quả False.

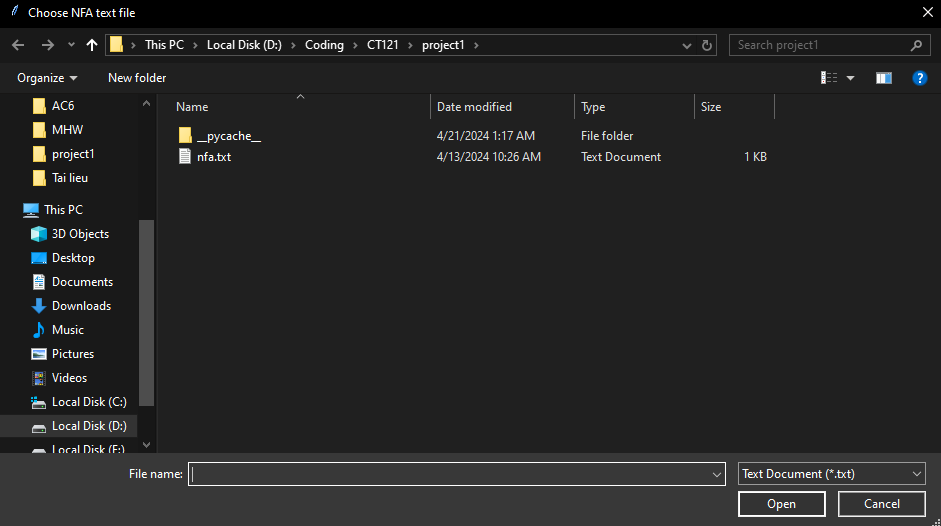
## 3. Giao diện hệ thống

* Giao diện khi chạy chương trình: Bao gồm một tiêu đề giới thiệu và 1 nút bấm xử lý sự kiện đọc file



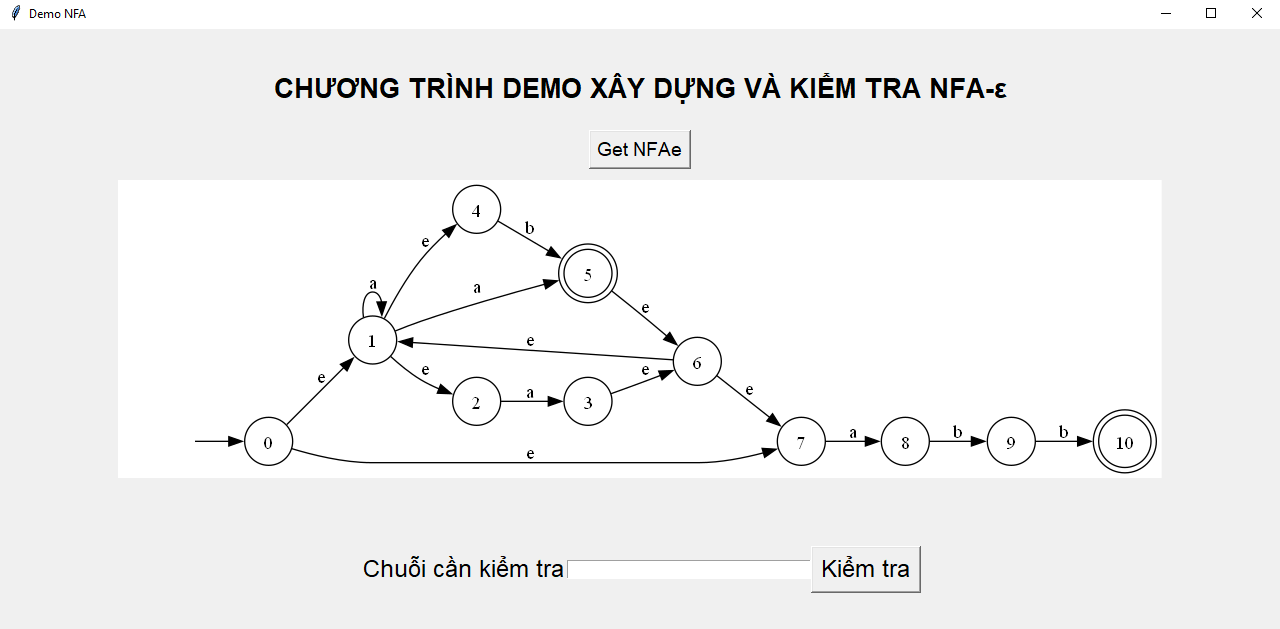
Hình 12. Giao diện khi chạy chương trình

* Hộp thoại yêu cầu người dùng chọn tập tin văn bản đầu vào



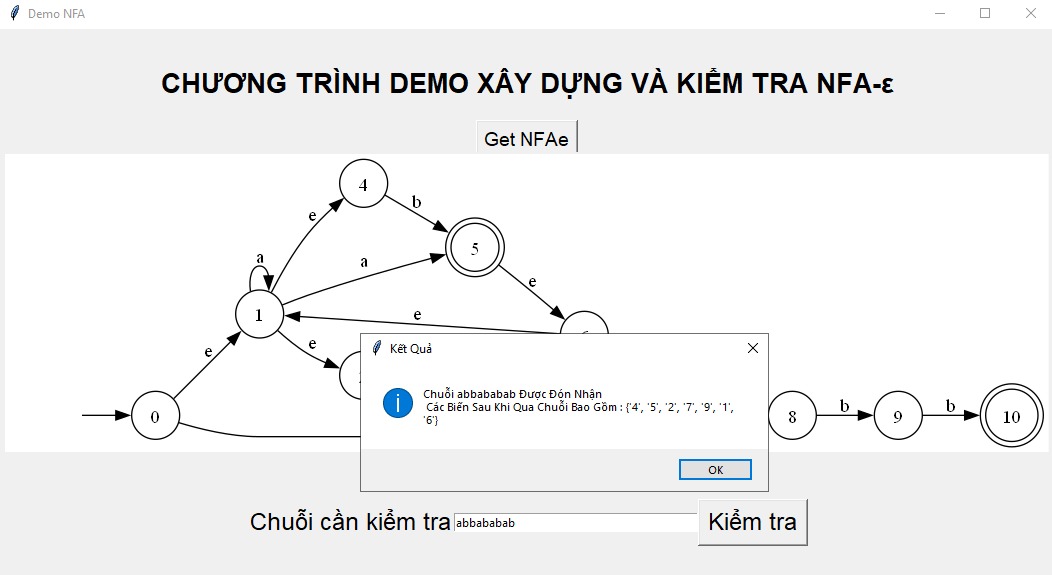
Hình 13. Giao diện khi người chọn tập tin đầu vào

* Giao diện chương trình sau khi người dùng đã chọn tập tin đầu vào: chương trình sẽ hiển thị hình ảnh mô tả NFA-ε ở dạng đồ thị được xây dựng từ tập tin người dùng vừa chọn, và hiển thị thêm một Input Field để người dùng có thể nhập vào một chuỗi và dùng nút “Kiểm tra” để xử lý sự kiện kiểm tra chuỗi



Hình 14. Giao diện khi người dùng đã chọn xong tập tin đầu vào

* Chương trình hiển thị kết quả kiểm tra chuỗi:



Hình 15. Giao diện kết quả kiểm tra chuỗi

# CHƯƠNG 3

# KIỂM THỬ VÀ ĐÁNH GIÁ

## 1. Mục tiêu

* **Độ chính xác**: Đảm bảo rằng hệ thống xây dựng NFA-ε và kiểm tra chuỗi hoạt động đúng và chính xác theo các yêu cầu cụ thể. Các kết quả của quá trình kiểm tra chuỗi phải phản ánh chính xác trạng thái của chuỗi trong NFA-ε đã cho.
* **Hiệu suất**: Đảm bảo rằng hệ thống hoạt động một cách hiệu quả, với thời gian phản hồi nhanh chóng và mức độ tiêu tốn tài nguyên hợp lý.
* **Tính bảo mật**: Đảm bảo rằng hệ thống không có lỗ hổng bảo mật và bảo vệ thông tin cá nhân của người dùng. Kiểm tra tính an toàn và bảo mật của dữ liệu được nhập và xử lý bởi hệ thống.
* **Tương thích**: Đảm bảo rằng hệ thống hoạt động tốt trên các nền tảng và trình duyệt khác nhau, cũng như với các phiên bản phần mềm và thư viện phụ thuộc khác nhau.
* **Thiết kế sáng tạo**: Đánh giá tính sáng tạo và sự tiện ích của giao diện người dùng và trải nghiệm người dùng. Tìm cách cải thiện và tối ưu hóa giao diện để tạo ra trải nghiệm tốt nhất cho người dùng.

## 2. Nghi thức kiểm tra

* 1. Kiểm thử tính năng nhập dữ liệu đầu vào từ file:
     1. Mục tiêu: nhằm kiểm tra tính đúng đắn và logic khi người dùng chọn file dữ liệu đầu vào.
     2. Kết nhập:

|  |  |  |
| --- | --- | --- |
| Mã số trường hợp | Nội dung nhập vào | Ghi chú |
| TH1\_01 | Không đầy đủ dữ liệu đầu vào | Trường hợp sai |
| TH1\_02 | Đầy đủ dữ liệu đầu vào | Truờng hợp đúng |

Bảng 1: Kết nhập tính năng nhập dữ liệu đầu vào từ file

* + 1. Kết xuất:

|  |  |  |  |
| --- | --- | --- | --- |
| Mã số trường hợp | Kết quả mong đợi | Kết quả thực tế | Ghi chú |
| TH1\_01 | Hệ thống thông báo “File không hợp lệ, vui lòng chọn file khác” | Như mong đợi | Thành công |
| TH1\_02 | Hệ thống sẽ chuyển tới nhập chuỗi kiểm tra | Như mong đợi | Thành công |

Bảng 2: Kết xuất tính năng nhập dữ liệu đầu vào từ file

* 1. Kiểm thử tính năng kiểm tra chuỗi nhập vào:
     1. Mục tiêu: nhằm kiểm tra tính đúng đắn và logic khi người dùng nhập chuỗi để kiểm tra.
     2. Kết nhập:

|  |  |  |
| --- | --- | --- |
| Mã số trường hợp | Nội dung nhập vào | Ghi chú |
| TH1\_01 | Đầy đủ và đúng thông tin trong các trường thông tin. | Trường hợp đúng |
| TH1\_02 | Đầy đủ và sai thông tin trong trường thông tin. | Truờng hợp sai |
| TH1\_03 | Để trống các trường nhập hoặc một trong các trường nhập thông tin. | Trường hợp sai |

Bảng 3: Kết nhập tính năng kiểm tra chuỗi nhập vào

* + 1. Kết xuất:

|  |  |  |  |
| --- | --- | --- | --- |
| Mã số trường hợp | Kết quả mong đợi | Kết quả thực tế | Ghi chú |
| TH1\_01 | Hệ thống thông báo “Chuỗi được đón nhận” | Như mong đợi | Thành công |
| TH1\_02 | Hệ thống thông báo “Chuỗi có kí tự không có. Vui lòng nhập lại ” | Như mong đợi | Thành công |
| TH1\_03 | Hệ thống thông báo “Chuỗi không được đón nhận” | Như mong đợi | Thành công |

Bảng 4: Kết xuất tính năng kiểm tra chuỗi nhập vào

# PHẦN KẾT LUẬN

## 1. Kết quả đạt được

Đạt được những yêu cầu nhất định về các chức năng cơ bản của đề tài như xây dựng từ NFA-ε từ tập tin văn bản, kiểm tra chuỗi, quản lý các bước dịch chuyển của ε

Ngoài ra, tạo được một giao diện đơn giản bằng thư viện tkinter và dùng thư viện automathon để hỗ trợ việc hiển thị hình ảnh mô tả NFA-ε

## 2. Hướng phát triển

* **Tối ưu hoá:** tối ưu hoá giải thuật quản lý NFA-ε, dịch chuyển ε và kiểm tra chuỗi để chương trình đạt được hiệu suất tốt hơn.
* **Mở rộng:** có thể mở rộng chương trình này bằng cách thêm vào các chức năng như tự động tối ưu NFA-ε, thiết kế các giải thuật chuyển NFA-ε sang NFA hoặc DFA tương đương.
* **Thiết kế giao diện người dùng:** thiết kế một giao diện đồ hoạ (GUI) trực quan và hoàn chỉnh để có thể sử dụng chương trình một cách chỉnh chu và phù hợp với những tính năng phát triển thêm về sau.
* **Tích hợp vào các ứng dụng thực tế:** nghiên cứu cách tích hợp chương trình vào các ứng dụng thực tế như trình biên dịch, công cụ phân tích ngôn ngữ, công cụ kiểm tra lỗi, ...

# TÀI LIỆU THAM KHẢO

[1] Giáo trình học phần “Tin học lý thuyết” (2009), cô Võ Huỳnh Trâm, Trường CNTT & TT - Trường ĐHCT.

[2] Slide bài giảng học phần “Tin học lý thuyết”, cô Phạm Xuân Hiền.