CHƯƠNG III

ÔTÔMÁT HỮU HẠN VÀ BIỂU THỨC CHÍNH QUY

Nội dung chính: Trong chương này, ta sẽ nghiên cứu một loại "máy trừu tượng" gọi là ôtômát hữu hạn. Chúng là công cụ dùng đoán nhận một lớp ngôn ngữ khá đơn giản gọi là lớp ngôn ngữ chính quy. Trước hết, hai dạng của ôtômát hữu hạn sẽ lần lượt được trình bày và có sự chứng minh rằng chúng tương đương nhau về khả năng đoán nhận ngôn ngữ. Tiếp đó, ta sẽ đề cập đến biểu thức chính quy - một phương tiện khác để xác định ngôn ngữ và ta lại thấy rằng lớp ngôn ngữ do các ôtômát hữu hạn chấp nhận chính là lớp ngôn ngữ chính quy. Phần tiếp theo của chương sẽ đề cập đến mối quan hệ giữa cơ chế ôtômát và các biểu thức chính quy dùng ký hiệu cho ngôn ngữ. Cuối chương này, một vài ứng dụng cụ thể của ôtômát hữu hạn sẽ được trình bày.

Mục tiêu cần đạt: Kết thúc chương này, sinh viên cần nắm vững :

- Khái niệm ôtômát hữu hạn, các thành phần, các dạng và sự khác biệt cơ bản giữa hai dạng.
- Cách thức chuyển đổi tương đương từ dạng đơn định sang không đơn định và ngược lại.
- Viết biểu thức chính quy ký hiệu cho tập ngôn ngữ chính quy.
- Mối liên quan giữa ôtômát hữu hạn và biểu thức chính quy.
- Vẽ sơ đồ chuyển trạng thái (đơn định hoặc không đơn định) từ một biểu thức chính quy.
- Tìm các ứng dụng thực tế khác từ mô hình ôtômát hữu hạn.

Kiến thức cơ bản: Để tiếp thu tốt nội dung của chương này, sinh viên cần có một số các kiến thức liên quan về lý thuyết đồ thị, lý thuyết mạch; hiểu các khái niệm cơ bản về kiến trúc máy tính; có sử dụng qua một số trình soạn thảo văn bản thông thường …

Tài liệu tham khảo :

[1] John E. Hopcroft, Jeffrey D.Ullman – *Introduction to Automata Theory, Languages and Computation* – Addison – Wesley Publishing Company, Inc – 1979 ***(Chapter 2 : Finite Automata and Regular Expressions).***

[2] Phan Thị Tươi – *Trình biên dịch* – Nhà xuất bản Giáo dục – 1986 ***(Chương 3 : Bộ phân tích từ vựng).***

[3] J.A.Garcia *and* S.Moral- *Theory of Finite Automata :*
http://decsai.ugr.es/~jags/fat.html

[4] Donald R. Biggar - *Regular Expression Matching Using Finite Automata:*
http://www3.sympatico.ca/dbiggar/FA.home.html

I. ÔTÔMÁT HỮU HẠN (FA : Finite Automata)

Ôtômát hữu hạn FA là một mô hình tính toán của hệ thống với sự mô tả bởi các input và output. Tại mỗi thời điểm, hệ thống có thể được xác định ở một trong số hữu hạn các cấu hình nội bộ gọi là các *trạng thái* (states). Mỗi trạng thái của hệ thống thể hiện sự tóm tắt các thông tin liên quan đến những input đã chuyển qua và xác định các phép chuyển kế tiếp trên dãy input tiếp theo.

Trong khoa học máy tính, ta có thể tìm thấy nhiều ví dụ về hệ thống trạng thái hữu hạn, và lý thuyết về ôtômát hữu hạn là một công cụ thiết kế hữu ích cho các hệ thống này. Chẳng hạn, một hệ chuyển mạch như bộ điều khiển (Control Unit) trong máy tính. Một chuyển mạch thì bao gồm một số hữu hạn các cổng (gate) input, mỗi cổng có 2 giá trị 0 hoặc 1. Các giá trị đầu vào này sẽ xác định 2 mức điện thế khác nhau ở cổng output. Mỗi trạng thái của một mạng chuyển mạch với n cổng bất kỳ sẽ là một trường hợp trong 2^n phép gán của 0 và 1 đối với các cổng khác nhau. Các chuyển mạch thì được thiết kế theo cách này, vì thế chúng có thể được xem như hệ thống trạng thái hữu hạn. Các chương trình sử dụng thông thường, chẳng hạn trình soạn thảo văn bản hay bộ phân tích từ vựng trong trình biên dịch máy tính cũng được thiết kế như các hệ thống trạng thái hữu hạn. Ví dụ bộ phân tích từ vựng sẽ quét qua tất cả các dòng ký tự của chương trình máy tính để tìm nhóm các chuỗi ký tự tương ứng với một tên biến, hằng số, từ khóa, …Trong quá trình xử lý này, bộ phân tích từ vựng cần phải nhớ một số hữu hạn thông tin như các ký tự bắt đầu hình thành những chuỗi từ khóa. Lý thuyết về ôtômát hữu hạn thường được dùng đến nhiều cho việc thiết kế các công cụ xử lý chuỗi hiệu quả.

Máy tính cũng có thể được xem như một hệ thống trạng thái hữu hạn. Trạng thái hiện thời của bộ xử lý trung tâm, bộ nhớ trong và các thiết bị lưu trữ phụ ở mỗi thời điểm bất kỳ là một trong những số rất lớn và hữu hạn của số trạng thái. Bộ não con người cũng là một hệ thống trạng thái hữu hạn, vì số các tế bào thần kinh hay gọi là neurons là số có giới hạn, nhiều nhất có thể là 2^{35}.

Lý do quan trọng nhất cho việc nghiên cứu các hệ thống trạng thái hữu hạn là tính tự nhiên của khái niệm và khả năng ứng dụng đa dạng trong nhiều lĩnh vực thực tế. Ôtômát hữu hạn (FA) được chia thành 2 loại: đơn định (DFA) và không đơn định (NFA). Cả hai loại ôtômát hữu hạn đều có khả năng nhận dạng chính xác tập chính quy. Ôtômát hữu hạn đơn định có khả năng nhận dạng ngôn ngữ dễ dàng hơn ôtômát hữu hạn không đơn định, nhưng thay vào đó thông thường kích thước của nó lại lớn hơn so với ôtômát hữu hạn không đơn định tương đương.

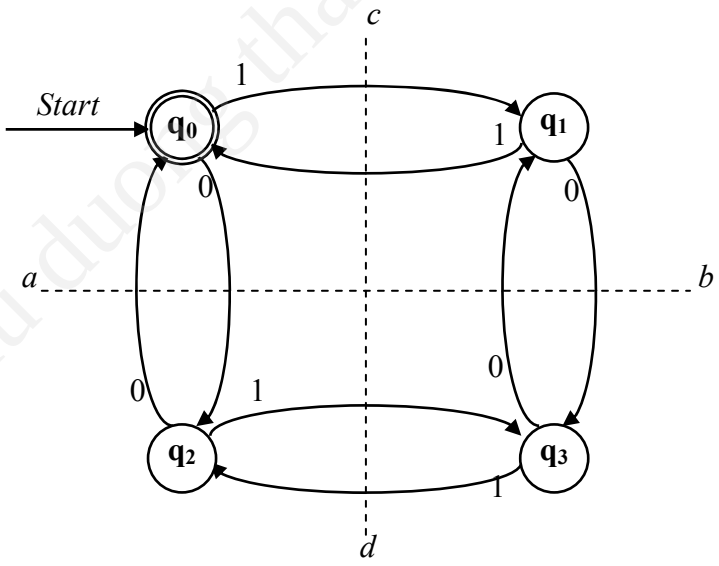
1.1. Ôtômát hữu hạn đơn định - DFA (Deterministic Finite Automata)

Một ôtômát hữu hạn đơn định (DFA) - gọi tắt là FA -gồm một tập hữu hạn các trạng thái và một tập các phép chuyển từ trạng thái này tới trạng thái khác trên các ký hiệu nhập (input symbols) được chọn từ một bộ chữ cái Σ nào đó. Mỗi ký hiệu nhập có đúng một phép chuyển khỏi mỗi trạng thái (có thể chuyển trở về chính nó). Một trạng thái, thường ký hiệu là q_0, gọi là trạng thái bắt đầu (trạng thái ôtômát bắt đầu). Một số trạng thái được thiết kế như là các trạng thái kết thúc hay trạng thái chấp nhận.

Một đồ thị có hướng, gọi là sơ đồ chuyển (transition diagram) tương ứng với một DFA như sau: các đỉnh của đồ thị là các trạng thái của DFA; nếu có một đường chuyển từ trạng thái q đến trạng thái p trên input a thì có một cung nhãn a chuyển từ trạng thái q đến trạng thái p trong sơ đồ chuyển. DFA chấp nhận một chuỗi x nếu như tồn tại dãy các phép chuyển tương ứng trên mỗi ký hiệu của x dẫn từ trạng thái bắt đầu đến một trong những trạng thái kết thúc.

Chẳng hạn, sơ đồ chuyển của một DFA được mô tả trong hình 3.1. Trạng thái khởi đầu q_0 được chỉ bằng mũi tên có nhãn "Start". Chỉ có duy nhất một trạng thái kết thúc, cũng là q_0 trong trường hợp này, được chỉ ra bằng hai vòng tròn. Ôtômát này chấp nhận tất cả các chuỗi số 0 và số 1 với số số 0 và số số 1 là số chẵn.

Thí dụ 3.1 :



Hình 3.1 - Sơ đồ chuyển của một DFA

Một điều cần lưu ý, DFA sử dụng mỗi trạng thái của nó để giữ chỉ một phần của chuỗi số 0 và 1 chứ không phải chứa một số thực sự, vì thế DFA cần dùng một số hữu hạn trạng thái.

Định nghĩa

Một cách hình thức ta định nghĩa ôtômát hữu hạn là bộ gồm năm thành phần (Q, Σ, δ, q_0, F), trong đó :

. Q là tập hợp hữu hạn các trạng thái.

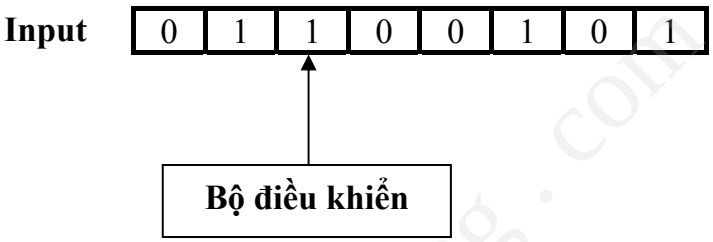
. Σ là bộ chữ cái nhập hữu hạn.

. δ là hàm chuyển ánh xạ từ $Q \times \Sigma \rightarrow Q$, tức là δ(q, a) là một trạng thái được cho bởi phép chuyển từ trạng thái q trên ký hiệu nhập a.

. $q_0 \in Q$ là trạng thái bắt đầu

. $F \subseteq Q$ là tập các trạng thái kết thúc.

Ta vẽ DFA như là bộ điều khiển hữu hạn, với mỗi trạng thái thuộc Q, DFA đọc một chuỗi các ký hiệu a từ Σ viết trên băng (như hình vẽ).



Hình 3.2 - Mô tả một DFA

Trong một lần chuyển, DFA đang ở trạng thái q đọc ký hiệu nhập a trên băng, chuyển sang trạng thái được xác định bởi hàm chuyển δ(q, a), rồi dịch đầu đọc sang phải một ký tự. Nếu δ(q, a) chuyển đến một trong những trạng thái kết thúc thì DFA chấp nhận chuỗi được viết trên băng input phía trước đầu đọc, nhưng không bao gồm ký tự tại vị trí đầu đọc vừa dịch chuyển đến. Trong trường hợp đầu đọc đã dịch đến cuối chuỗi trên băng, thì DFA mới chấp nhận toàn bộ chuỗi trên băng.

Hàm chuyển trạng thái mở rộng

Để có thể mô tả một cách hình thức hoạt động của một DFA trên chuỗi, ta mở rộng hàm chuyển δ để áp dụng đối với một trạng thái trên chuỗi hơn là một trạng thái trên từng ký hiệu. Ta định nghĩa hàm chuyển δ như một ánh xạ từ $Q \times \Sigma^* \rightarrow Q$ với ý nghĩa δ(q, w) là trạng thái DFA chuyển đến từ trạng thái q trên chuỗi w. Một cách hình thức, ta định nghĩa :

1. $\delta(q, \varepsilon) = q$
2. $\delta(q, wa) = \delta(\delta(q, w), a)$, với mọi chuỗi w và ký hiệu nhập a.

Một số quy ước về ký hiệu :

- Q là tập các trạng thái. Ký hiệu q và p (có hoặc không có chỉ số) là các trạng thái, q_0 là trạng thái bắt đầu.
- Σ là bộ chữ cái nhập. Ký hiệu a, b (có hoặc không có chỉ số) và các chữ số là các ký hiệu nhập.
- δ là hàm chuyển.
- F là tập các trạng thái kết thúc.

- w, x, y và z (có hoặc không có chỉ số) là các chuỗi ký hiệu nhập.

Ngôn ngữ được chấp nhận bởi DFA

Một chuỗi w được chấp nhập bởi ôtômát hữu hạn M (Q, Σ, δ, q_0, F) nếu $\delta(q_0, w) = p$ với $p \in F$. Ngôn ngữ được chấp nhận bởi M, ký hiệu L(M) là tập hợp:

$$L(M) = \{ w \mid \delta(q_0, w) \in F \}$$

Thí dụ 3.2 : Xét sơ đồ chuyển ở hình 3.1. Theo khái niệm hình thức, ta có DFA được xác định bởi M (Q, Σ, δ, q_0, F) với $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{0, 1\}$, $F = \{q_0\}$ và hàm chuyển δ như sau:

δ	Inputs	
Trạng thái	0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Giả sử chuỗi w = 110101 được nhập vào M.
Ta có $\delta(q_0, 1) = q_1$ và $\delta(q_1, 1) = q_0$,vậy $\delta(q_0, 11) = \delta(\delta(q_0,1),1) = \delta(q_1, 1) = q_0$.
Tiếp tục $\delta(q_0, 0) = q_2$, vậy $\delta(q_0, 110) = \delta(\delta(q_0, 11), 0) = q_2$.
Tiếp tục ta có $\delta(q_0, 1101) = q_3$, $\delta(q_0, 11010) = q_1$
Và cuối cùng $\delta(q_0, 110101) = q_0 \in F$.
(Hay $\delta(q_0, 110101) = \delta(q_1, 10101) = \delta(q_0, 0101) = \delta(q_2, 101) = \delta(q_3, 01) = \delta(q_1, 1) = q_0 \in F$)

Vậy 110101 thuộc L(M). Ta có thể chứng minh rằng L(M) là tập mọi chuỗi có số chẵn số 0 và số chẵn số 1.

Theo mô tả DFA như trên, ta thấy cũng có thể dùng *bảng hàm chuyển* (transition table) để mô tả các phép chuyển trạng thái của một ôtômát hữu hạn. Trong bảng hàm chuyển, hàng chứa các trạng thái thuộc tập trạng thái của ôtômát và cột là các ký hiệu thuộc bộ chữ cái nhập. Bảng hàm chuyển gợi ý cho chúng ta một cấu trúc dữ liệu để mô tả cho một ôtômát hữu hạn, đồng thời cũng cho thấy có thể dễ dàng mô phỏng hoạt động của DFA thông qua một chương trình máy tính, chẳng hạn dùng cấu trúc vòng lặp.

Giải thuật mô phỏng hoạt động của một DFA

```
. Input : Chuỗi nhập x kết thúc bởi $
. Output : Câu trả lời "YES" nếu DFA chấp nhận chuỗi x và "NO" nếu
ngược lại.
. Giải thuật :
        q := q0 ;
        c := nextchar ;          { c là ký hiệu nhập được đọc tiếp theo }
        While c <> $ do
                begin
                        q := δ(q, c);
                        c := nextchar ;
                end
```

Nhận xét :

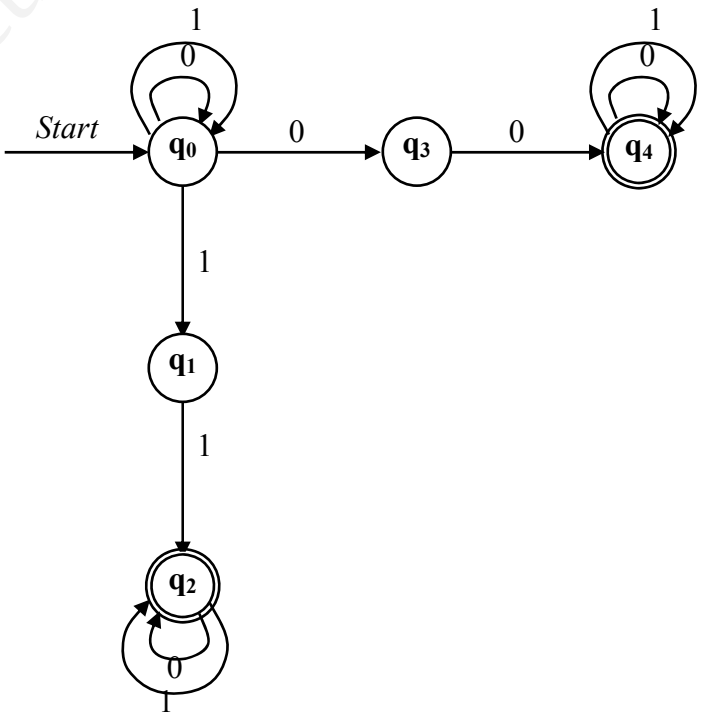
Một cách tổng quát, ta thấy tập Q của DFA thể hiện các trạng thái lưu trữ của ôtômát trong quá trình đoán nhận ngôn ngữ, và như vậy khả năng lưu trữ của ôtômát là hữu hạn. Mặt khác, hàm chuyển δ là hàm toàn phần và đơn trị, cho nên các bước chuyển của ôtômát luôn luôn được xác định một cách duy nhất. Chính vì hai đặc điểm này mà DFA mô tả như trên được gọi là ôtômát hữu hạn đơn định.

1.2. Ôtômát hữu hạn không đơn định - NFA (Nondeterministic Finite Automata)

Xét một dạng sửa đổi mô hình DFA để chấp nhận không, một hoặc nhiều hơn một phép chuyển từ một trạng thái trên cùng một ký hiệu nhập. Mô hình mới này gọi là ôtômát hữu hạn không đơn định (NFA).

Một chuỗi ký hiệu nhập $a_1\ a_2\ ...\ a_n$ được chấp nhận bởi một NFA nếu có tồn tại một chuỗi các phép chuyển, tương ứng với chuỗi nhập, từ trạng thái bắt đầu đến trạng thái kết thúc. Chẳng hạn, chuỗi 01001 được chấp nhận bởi ôtômát trong hình dưới đây vì có chuỗi phép chuyển qua các trạng thái $q_0, q_0, q_0, q_3, q_4, q_4$ có nhãn tương ứng là 0, 1, 0, 0, 1. NFA này chấp nhận tất cả các chuỗi có hai số 0 liên tiếp hoặc hai số 1 liên tiếp.

Thí dụ 3.3 :

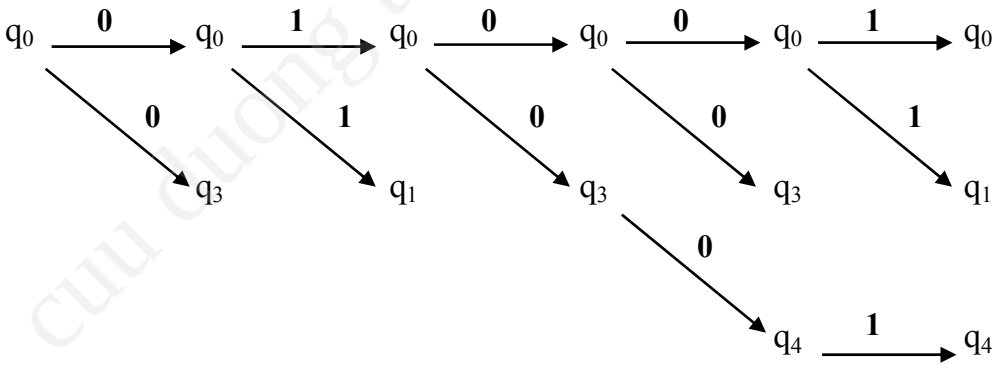


Hình 3.3 - Sơ đồ chuyển của một NFA

Chú ý rằng có thể xem ôtômát hữu hạn đơn định - DFA (hay gọi tắt là FA) là một trường hợp đặc biệt của NFA, trong đó mỗi trạng thái chỉ có duy nhất một phép chuyển trên mỗi ký hiệu nhập. Vì thế trong DFA, với một chuỗi nhập w và trạng thái q, chỉ có đúng một đường đi nhãn w bắt đầu từ q. Để xác định chuỗi w có được chấp nhận bởi DFA hay không chỉ cần kiểm tra đường đi này. Nhưng đối với NFA, có thể có nhiều đường đi có nhãn là w, và do đó tất cả phải được kiểm tra để thấy có hay không có đường đi tới trạng thái kết thúc.

Tương tự như DFA, NFA cũng hoạt động với một bộ điều khiển hữu hạn đọc trên băng nhập. Tuy nhiên, tại mỗi thời điểm, bộ điều khiển có thể chứa một số bất kỳ trạng thái. Khi có sự lựa chọn trạng thái kế tiếp, chẳng hạn như từ trạng thái q_0 trên ký hiệu nhập 0 ở hình 3.3, ta phải tưởng tượng như có các bản sao của ôtômát đang thực hiện đồng thời. Mỗi trạng thái kế tiếp mà ôtômát có thể chuyển đến sẽ tương ứng với một bản sao của ôtômát mà tại đó bộ điều khiển đang chứa trạng thái đó.

Chẳng hạn, với chuỗi 01001, ta có :



Định nghĩa

Một cách hình thức ta định nghĩa ôtômát hữu hạn không đơn định NFA là một bộ 5 thành phần $(Q, \Sigma, \delta, q_0, F)$ trong đó Q, Σ, q_0 và F có ý nghĩa như trong DFA, nhưng δ là hàm chuyển ánh xạ từ $Q \times \Sigma \to 2^Q$.
Khái niệm $\delta(q, a)$ là tập hợp tất cả các trạng thái p sao cho có phép chuyển trên nhãn a từ trạng thái q tới p.

Hàm chuyển trạng thái mở rộng

Để thuận tiện trong việc mô tả hoạt động ôtômát trên chuỗi, ta mở rộng hàm chuyển δ ánh xạ từ $Q \times \Sigma^* \to 2^Q$ như sau :

1. $\delta(q, \varepsilon) = \{q\}$
2. $\delta(q, wa) = \{ p \mid$ có một trạng thái r trong $\delta(q, w)$ mà p thuộc $\delta(r, a)\}$
   $= \delta(\delta(q, w), a)$
3. $\delta(P, w) = \cup_{q \in P} \delta(q, w)$, $\forall P \subseteq Q$.

Ngôn ngữ được chấp nhận bởi NFA

Ngôn ngữ L(M), với M là ôtômát hữu hạn không đơn định NFA $(Q, \Sigma, \delta, q_0, F)$ là tập hợp :
$L(M) = \{w \mid \delta(q_0, w)$ có chứa một trạng thái trong F $\}$
Thí dụ 3.4 : Xét sơ đồ chuyển của hình 3.3. Theo khái niệm hình thức, ta có :
NFA M $(\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \delta, q_0, \{q_2, q_4\})$ với hàm chuyển δ như sau :

δ	Inputs	
Trạng thái	0	1
q_0	$\{q_0,q_3\}$	$\{q_0,q_1\}$
q_1	∅	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_2\}$
q_3	$\{q_4\}$	∅
q_4	$\{q_4\}$	$\{q_4\}$

Xét chuỗi nhập w = 01001
Ta có : $\delta(q_0, 0) = \{q_0, q_3\}$
$\delta(q_0, 01) = \delta(\delta(q_0, 0),1) = \delta(\{q_0, q_3\},1) = \delta(q_0, 1) \cup \delta(q_3, 1) = \{q_0, q_1\}$
Tương tự , ta có thể tính :
$\delta(q_0, 010) = \{q_0, q_3\}$
$\delta(q_0, 0100) = \{q_0, q_3, q_4\}$
và $\delta(q_0, 01001) = \{q_0, q_1, q_4\}$
Do $q_4 \in F$ nên $w \in L(M)$.

Câu hỏi :

1. Hãy cho nhận xét về điểm khác biệt quan trọng giữa DFA và NFA ?
2. Theo bạn, dạng đơn định hay không đơn định sẽ dùng nhận dạng một chuỗi dễ dàng hơn ?

1.3. Sự tương đương giữa DFA và NFA

Vì mỗi DFA là một NFA, nên rõ ràng lớp ngôn ngữ được chấp nhận bởi NFA cũng bao gồm các tập chính quy (đây chính là ngôn ngữ được chấp nhận bởi DFA). Tuy nhiên, không có cơ sở để nói rằng NFA chỉ chấp nhận duy nhất các tập hợp này. Điều đó cho thấy DFA có thể mô phỏng được hoạt động của NFA, nghĩa là với mỗi NFA,

ta có thể xây dựng một DFA tương đương (chấp nhận cùng một ngôn ngữ với nó). Đặt một DFA mô phỏng hoạt động của NFA là cho phép các trạng thái của DFA tương ứng với tập các trạng thái của NFA. Tại mỗi thời điểm, DFA lưu giữ trong bộ điều khiển tất cả các trạng thái mà NFA có thể chuyển đến khi đọc cùng một input như DFA.

ĐỊNH LÝ 3.1 : Nếu L là tập được chấp nhận bởi một NFA thì tồn tại một DFA chấp nhận L.

Chứng minh

Đặt M ($Q, \Sigma, \delta, q_0, F$) là NFA chấp nhận L.

Ta xây dựng DFA M' ($Q', \Sigma, \delta', q_0', F'$) tương đương như sau:

- Các trạng thái của M' là tất cả các tập hợp con của tập trạng thái của M, hay $Q' = 2^Q$. Tại mỗi thời điểm, M' sẽ lưu giữ trong trạng thái của nó tất cả các trạng thái có thể của M. Một phần tử trong Q' được ký hiệu là $[q_1, q_2,..., q_i]$, trong đó các trạng thái $q_1, q_2,..., q_i \in Q$. Ta xem $[q_1, q_2,..., q_i]$ là một trạng thái đơn của DFA tương ứng với một tập trạng thái của NFA.
- $q_0' = [q_0]$.
- F' là tập hợp các trạng thái của Q' có chứa ít nhất một trạng thái kết thúc trong tập F của M.
- Ta định nghĩa hàm chuyển δ' như sau :

$\delta'([q_1, q_2,..., q_i], a) = [p_1, p_2,..., p_j]$ nếu và chỉ nếu $\delta(\{q_1, q_2,..., q_i\}, a) = \{p_1, p_2,..., p_j\}$

Bây giờ ta chứng minh quy nạp theo độ dài của chuỗi nhập x rằng:

$$\delta'(q_0', x) = [q_1, q_2,..., q_i] \Leftrightarrow \delta(q_0, x) = \{q_1, q_2,..., q_i\} \quad (1)$$

Với $|x| = 0$, ta có $x = \varepsilon$ và $q_0' = [q_0]$ nên (1) hiển nhiên đúng

Giả sử (1) đúng với các chuỗi nhập có độ dài tới m.

Xét chuỗi nhập có độ dài m + 1, đặt chuỗi này là xa với $a \in \Sigma$, ta có :

$$\delta'(q_0', xa) = \delta'(\delta'(q_0', x), a)$$

Theo định nghĩa :

$\delta'([p_1, p_2,..., p_i], a) = [r_1, r_2,..., r_k] \Leftrightarrow \delta(\{p_1, p_2,..., p_j\}, a) = \{r_1, r_2,..., r_k\}$.

Mặt khác theo giả thiết quy nạp $\delta'(q_0', x) = [p_1, p_2,..., p_j] \Leftrightarrow \delta(q_0, x) = \{p_1, p_2,..., p_j\}$, nên thay vào ta có : $\delta'(q_0', xa) = [r_1, r_2,..., r_k] \Leftrightarrow \delta(q_0, xa) = \{r_1, r_2,..., r_k\}$.

Dễ thấy rằng $\delta'(q_0', x) \in F'$ khi và chỉ khi $\delta(q_0, x)$ có chứa ít nhất một trạng thái $\in F$.

Vậy L(M) = L(M')

Vì NFA và DFA chấp nhận cùng các tập hợp, nên ta sẽ không phân biệt chúng trừ khi điều đó thật sự cần thiết, sẽ đơn giản hơn để hiểu cả hai cùng là ôtômát đơn định.

Thí dụ 3.5 : Cho NFA M ($\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\}$) với hàm chuyển δ như sau :

$\delta(q_0, 0) = \{q_0, q_1\}$, $\delta(q_0, 1) = \{q_1\}$, $\delta(q_1, 0) = \varnothing$, $\delta(q_1, 1) = \{q_0, q_1\}$

Ta xây dựng DFA tương đương M' ($Q', \{0, 1\}, \delta', [q_0], F'$) chấp nhận L(M) như sau :

. Q' : chứa tất cả các tập con của $\{q_0, q_1\}$, vậy $Q' = \{[q_0], [q_1], [q_0, q_1], \varnothing\}$

. Hàm chuyển δ' :

Vì $\delta(q_0, 0) = \{q_0, q_1\}$ nên $\delta'([q_0], 0) = [q_0, q_1]$
Tương tự : $\delta'([q_0], 1) = [q_1]$
$\delta'([q_1], 0) = \varnothing$
$\delta'([q_1], 1) = [q_0, q_1]$
Mặt khác : $\delta'(\varnothing, 0) = \delta'(\varnothing, 1) = \varnothing$
Cuối cùng : $\delta'([q_0, q_1],0) = [q_0, q_1]$
(vì $\delta(\{q_0, q_1\},0) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \varnothing = \{q_0, q_1\}$)
$\delta'([q_0, q_1], 1) = [q_0, q_1]$
(vì $\delta(\{q_0, q_1\},1) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_1\} \cup \{q_0, q_1\} = \{q_0, q_1\}$)
. Tập trạng thái kết thúc $F' = \{[q_1], [q_0, q_1]\}$

Thực tế, có rất nhiều trạng thái của NFA không có hàm chuyển đến từ trạng thái bắt đầu $[q_0]$. Do đó, thông thường, cách tốt nhất là ta nên xây dựng DFA tương đương bắt đầu từ trạng thái $[q_0]$ và thêm vào các trạng thái mới cho DFA chỉ khi có các hàm chuyển từ một trạng thái đã được thêm vào trước đó.

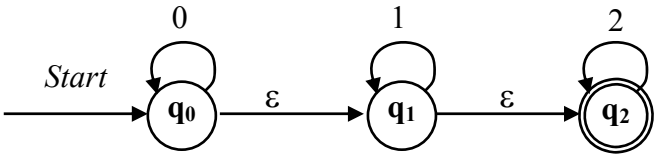
Câu hỏi :

Bạn có nhận xét gì về kích thước giữa một DFA và một NFA tương đương với nó chấp nhận cùng một tập ngôn ngữ ?

1.4. NFA với ε-dịch chuyển (NFAε)

Ta mở rộng mô hình NFA cho phép các phép chuyển trên nhãn rỗng ε. Sơ đồ chuyển sau đây của một NFA chấp nhận chuỗi gồm một số bất kỳ (có thể là 0) chữ số 0 sau đó là một số bất kỳ chữ số 1 và sau nữa là một số bất kỳ chữ số 2. Thông thường, ta nói NFA chấp nhận một chuỗi w nếu có đường truyền nhãn w từ trạng thái bắt đầu đến một trạng thái kết thúc. Chẳng hạn, chuỗi 002 được chấp nhận bởi đường truyền $q_0, q_0, q_0, q_1, q_2, q_2$ với các cung nhãn 0, 0, ε, ε, 2.

Thí dụ 3.6 : Sơ đồ chuyển của một NFA với ε-dịch chuyển :



Hình 3.4 - *NFA với ε-dịch chuyển*

Định nghĩa: Một cách hình thức ta định nghĩa NFA với ε-dịch chuyển là bộ 5 thành phần $(Q, \Sigma, \delta, q_0, F)$ với tất cả các thành phần có ý nghĩa như trên, nhưng hàm chuyển δ là ánh xạ từ $Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$.

Khái niệm δ(q, a) gồm tất cả các trạng thái p sao cho có phép chuyển nhãn a từ q tới p, trong đó a là một ký hiệu thuộc Σ hoặc là ε.

Hàm chuyển trạng thái mở rộng: Ta mở rộng hàm chuyển δ thành hàm chuyển δ^* ánh xạ từ $Q \times \Sigma^* \to 2^Q$. $\delta^*(q,w)$ chứa tất cả các trạng thái p sao cho có thể đi từ q tới p theo đường đi nhãn w (có thể chứa cạnh nhãn ε).

Ta sử dụng ε-CLOSURE(q) để xác định tập tất cả các đỉnh p sao cho có đường đi từ q tới p với nhãn ε.

Thí dụ 3.7 : Trong hình 3.4, ε-CLOSURE(q_0) = {q_0, q_1, q_2}.

Vì đường đi chỉ có một đỉnh q_0 (không có cung trên đường đi) là đường đi từ q_0 tới q_0 có tất cả các cạnh nhãn là ε. Đường đi q_0, q_1 chỉ ra rằng q_1 thuộc ε-CLOSURE(q_0). Và đường đi q_0, q_1, q_2 chỉ ra rằng q_2 thuộc ε-CLOSURE(q_0).

Đặt **ε-CLOSURE(P)** = $\cup_{q \in P}$ **ε-CLOSURE(q)**, trong đó P là một tập các trạng thái và q là một trạng thái. Ta định nghĩa hàm δ^* như sau:

1. $\delta^*(q, \varepsilon)$ = ε-CLOSURE(q)
2. $\delta^*(q, wa)$ = ε-CLOSURE(P),
   trong đó tập P = {p | có r trong $\delta^*(q, w)$ sao cho $p \in \delta(r, a)$}, $\forall w \in \Sigma^*$ và $a \in \Sigma$
   Hay $\delta^*(q, wa)$ = ε-CLOSURE($\delta(\delta^*(q, w), a)$

Ta mở rộng δ và δ^* trên tập hợp các trạng thái R như sau :

3. $\delta(R, a) = \cup_{q \in R} \delta(q, a)$, và
4. $\delta^*(R, w) = \cup_{q \in R} \delta^*(q, w)$

Câu hỏi :

Hãy so sánh sự khác biệt giữa hàm chuyển δ và δ^* ?

Nhận xét : $\delta^*(q, a)$ và δ(q, a) không nhất thiết bằng nhau vì $\delta^*(q, a)$ gồm tất cả các trạng thái có thể chuyển đến được từ q trên nhãn a gồm cả đường đi nhãn ε, trong khi đó δ(q, a) chỉ gồm các trạng thái có thể đến được từ q chỉ bằng các cung nhãn a. Tương tự $\delta^*(q, \varepsilon)$ có thể cũng không bằng δ(q, ε). Vì vậy ta phải phân biệt ký hiệu δ và δ^* đối với NFA với ε-dịch chuyển.

Ngôn ngữ được chấp nhận bởi NFAε:

Ta định nghĩa L(M), ngôn ngữ được chấp nhận bởi NFAε M = (Q, Σ, δ, q_0, F) là tập hợp các chuỗi :

$$L(M) = \{w \mid \delta^*(q_0, w) \text{ có chứa ít nhất một trạng thái trong } F\}$$

Thí dụ 3.8 : Xét sơ đồ chuyển của hình 3.4.

Theo khái niệm hình thức, ta có NFA M ($\{q_0, q_1, q_2\}$, {0, 1, 2}, δ, q_0, $\{q_2\}$) với hàm chuyển δ như sau :

δ	Inputs			
Trạng thái	0	1	2	ε
q_0	$\{q_0\}$	∅	∅	$\{q_1\}$
q_1	∅	$\{q_1\}$	∅	$\{q_1\}$
q_2	∅	∅	$\{q_2\}$	∅

Xét chuỗi nhập w = 012.
Ta cần tính $\delta^*(q_0, 012)$

Ta có : $\delta^*(q_0, \varepsilon) = \varepsilon\text{-CLOSURE}(q_0) = \{q_0, q_1, q_2\}$

vậy $\delta^*(q_0, 0) = \varepsilon\text{-CLOSURE}(\delta(\delta^*(q_0, \varepsilon), 0)$
$= \varepsilon\text{-CLOSURE}(\delta(\{q_0, q_1, q_2\}, 0))$
$= \varepsilon\text{-CLOSURE}(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0))$
$= \varepsilon\text{-CLOSURE}(\{q_0\} \cup \varnothing \cup \varnothing)$
$= \varepsilon\text{-CLOSURE}(\{q_0\}) = \{q_0, q_1, q_2\}$

và $\delta^*(q_0, 01) = \varepsilon\text{-CLOSURE}(\delta(\delta^*(q_0, 0), 1))$
$= \varepsilon\text{-CLOSURE}(\delta(\{q_0, q_1, q_2\}, 1))$
$= \varepsilon\text{-CLOSURE}(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1))$
$= \varepsilon\text{-CLOSURE}(\varnothing \cup \{q_1\} \cup \varnothing)$
$= \varepsilon\text{-CLOSURE}(\{q_1\}) = \{q_1, q_2\}$

$\Rightarrow \delta^*(q_0, 012) = \varepsilon\text{-CLOSURE}(\delta(\delta^*(q_0, 01), 2))$
$= \varepsilon\text{-CLOSURE}(\delta(\{q_1, q_2\}, 2))$
$= \varepsilon\text{-CLOSURE}(\delta(q_1, 2) \cup \delta(q_2, 2))$
$= \varepsilon\text{-CLOSURE}(\varnothing \cup \{q_2\})$
$= \varepsilon\text{-CLOSURE}(\{q_2\}) = \{\mathbf{q_2}\}$

Do $\delta^*(q_0, 012)$ có chứa trạng thái $q_2 \in F$ nên chuỗi $w \in L(M)$.

Giải thuật mô phỏng hoạt động của một NFAε :

. Input : Chuỗi nhập x được kết thúc bởi $.
. Output : Câu trả lời "YES" nếu NFA chấp nhận chuỗi x và "NO" nếu ngược lại.
. *Giải thuật :*

```
q := ε-CLOSURE(q0);
c := nextchar ;        { c là ký hiệu nhập được đọc tiếp theo }
While c <> $ do
        begin
                q := ε-CLOSURE(δ(q, c));
                c := nextchar ;
        end
```

If q in F **then** write ("YES") **else** write ("NO");

1.5. Sự tương đương giữa NFA có và không có ε-dịch chuyển

Tương tự như NFA, khả năng có thể thực hiện phép chuyển trên nhãn ε của NFAε cũng không làm cho NFAε chấp nhận được các tập hợp không chính quy. Ta có thể dẫn chứng điều này bằng cách mô phỏng hoạt động của một NFAε bởi một NFA không có ε-dịch chuyển.

ĐỊNH LÝ 3.2 : Nếu L được chấp nhận bởi một NFA có ε-dịch chuyển thì L cũng được chấp nhận bởi một NFA không có ε-dịch chuyển.

Chứng minh

Đặt M (Q, Σ, δ, q_0, F) là NFA với ε-dịch chuyển.

Ta xây dựng NFA M'(Q, Σ, δ', q_0, F') tương đương không có ε-dịch chuyển, trong đó:

. F' = $\begin{cases} F \cup \{q_0\} & \text{nếu } \varepsilon\text{-CLOSURE}(q_0) \text{ chứa một trạng thái thuộc F} \\ F & \text{trong các trường hợp còn lại} \end{cases}$

. δ'(q, a) là δ^*(q, a) với q ∈ Q và a ∈ Σ. Chú ý rằng M' không có ε-dịch chuyển nên ta có thể dùng δ' thay cho δ^*, nhưng phải phân biệt δ và δ^*.

Ta chứng minh bằng quy nạp trên | x | rằng $\delta'(q_0, x) = \delta^*(q_0, x)$. Tuy nhiên, điều đó có thể không đúng với $x = \varepsilon$ vì $\delta'(q_0, \varepsilon) = \{q_0\}$ trong khi $\delta^*(q_0, \varepsilon) = \varepsilon\text{-CLOSURE}(q_0)$. Do đó, cơ sở quy nạp bắt đầu với độ dài chuỗi là 1.

Với | x | = 1 thì x là một ký hiệu a và $\delta'(q, a) = \delta^*(q, a)$ theo định nghĩa δ'.

Xét | x | > 1: đặt x = wa với a là một ký hiệu trong Σ.

Ta có $\delta'(q, wa) = \delta'(\delta'(q_0, w), a)$

Theo giả thiết quy nạp thì $\delta'(q_0, w) = \delta^*(q_0, w)$. Đặt $\delta^*(q_0, w) = P$, ta cần chỉ ra rằng $\delta(P, a) = \delta^*(q_0, wa)$.

Ta có $\delta'(P, a) = \cup_{q \in P} \delta'(q, a) = \cup_{q \in P} \delta^*(q, a)$.

Hơn nữa vì $P = \delta^*(q_0, w)$ nên $\cup_{q \in P} \delta^*(q, a) = \delta^*(q_0, wa)$ (theo quy tắc 2 trong định nghĩa δ^*).

Vậy $\delta'(q_0, wa) = \delta^*(q_0, wa)$

Để đầy đủ chứng minh ta còn phải chỉ ra rằng $\delta'(q_0, x)$ chứa một trạng thái trong F' nếu và chỉ nếu $\delta^*(q_0, x)$ chứa một trạng thái trong F.

Nếu $x = \varepsilon$ thì điều đó hiển nhiên đúng (theo định nghĩa của F')

Nếu $x \neq \varepsilon$ thì ta đặt x = wa với $a \in \Sigma$.

Nếu $\delta^*(q_0, x)$ chứa một trạng thái trong F thì chắc chắn $\delta'(q_0, x)$ chứa cùng trạng thái trong F'. Ngược lại, nếu $\delta'(q_0, x)$ chứa một trạng thái trong F' khác hơn q_0 thì $\delta(q_0, x)$ phải chứa một trạng thái trong F (vì tập F và F' chỉ chênh lệch nhau trạng thái q_0). Nếu $\delta'(q_0, x)$ có chứa trạng thái q_0 và q_0 cũng là một trạng thái thuộc tập trạng thái kết thúc F thì vì $\delta^*(q_0, x) = \varepsilon\text{-CLOSURE}(\delta(\delta^*(q_0, w),a))$, nên trạng thái chung trong $\varepsilon\text{-CLOSURE}(q_0)$ và trong F phải ở trong $\delta^*(q_0, x)$.

Thí dụ 3.9 : Chuyển NFA với ε-dịch chuyển ở hình 3.4 sang dạng NFA không có chứa ε-dịch chuyển.

Ta xây dựng NFA tương đương M'(Q, Σ, δ', q_0, F') chấp nhận L(M) với các thành phần : . Q = {q_0, q_1, q_2}

. Σ = {0, 1, 2}

. Trạng thái bắt đầu : q_0

. F' = {q_0, q_2} do ε-CLOSURE(q_0) = {q_0, q_1, q_2} có chứa $q_2 \in F$

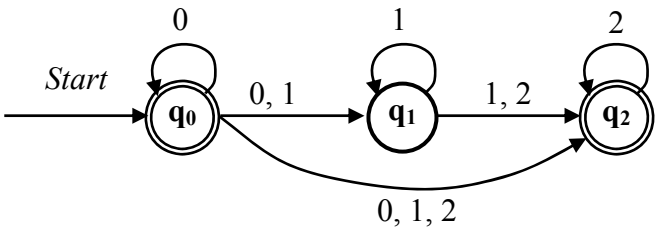
. Hàm chuyển δ' của M' được xác định theo công thức :

$$\delta'(q, a) = \delta^*(q, a) = \varepsilon\text{-CLOSURE}(\delta(\delta^*(q_0, \varepsilon), a)$$

Kết quả được chỉ ra trong bảng hàm chuyển sau :

δ'	Inputs		
Trạng thái	0	1	2
q_0	{q_0, q_1, q_2}	{q_1, q_2}	{q_2}
q_1	∅	{q_1, q_2}	{q_2}
q_2	∅	∅	{q_2}

Sơ đồ chuyển trạng thái:



Hình 3.5 - NFA tương đương cho thí dụ 3.9

1.6. Giải thuật xây dựng DFA từ NFA

Qua khảo sát các dạng mở rộng từ mô hình ôtômát hữu hạn ban đầu, ta thấy DFA thực chất là một trường hợp đặc biệt của NFA, nhưng :

- Nó không có sự truyền rỗng (truyền trên nhãn ε)

- Với mỗi trạng thái q và ký hiệu nhập a, chỉ có duy nhất một đường truyền đến một trạng thái khác.

Giả sử mỗi trạng thái của DFA là một tập trạng thái của NFA, DFA dùng trạng thái của mình để lưu giữ tất cả các trạng thái của NFA đạt được sau khi NFA đọc một ký tự nhập. Như vậy sau khi đọc các ký tự nhập a_1, a_2, ... , a_n, DFA ở trạng thái là tập con của các trạng thái thuộc NFA, đạt được khi NFA đi từ trạng thái bắt đầu theo một con đường nào đó có tên a_1a_2 ... a_n. Số trạng thái của DFA lúc đó phải bằng số phần tử trong tập lũy thừa của số trạng thái NFA. Song, trên thực tế trường hợp xấu nhất này ít khi xảy ra. Các trạng thái thực sự được dùng trong sơ đồ chuyển cho một DFA sẽ được xác định theo các phép chuyển trạng thái trên nhãn là mọi ký hiệu từ trạng thái bắt đầu của DFA, và sau đó lần lượt được bổ sung thêm vào tập trạng thái nếu như nó chưa có trong đó.

Giải thuật chi tiết được trình bày như sau :

Input: Một ôtômát hữu hạn không đơn định NFA.

Output: Một ôtômát hữu hạn đơn định DFA nhận dạng cùng ngôn ngữ như NFA.

Phương pháp: Xây dựng bảng hàm chuyển cho DFA mô phỏng đồng thời tất cả các chuyển dịch của NFA trên chuỗi nhập cho trước.

Ta dùng các tác vụ sau để lưu giữ các tập trạng thái của NFA :

(q : là một trạng thái của NFA, T : là tập trạng thái của NFA)

a) ε-closure(q) : là tập trạng thái của NFA đạt được từ trạng thái q trên sự truyền rỗng.

b) ε-closure(T) : là tập trạng thái của NFA đạt được từ tất cả các trạng thái q thuộc tập T trên sự truyền rỗng.

c) δ(T, a) : là tập trạng thái của NFA đạt được từ tất cả các trạng thái q thuộc tập T trên sự truyền bằng ký hiệu a.

Phân tích:

Trước khi đọc vào một ký tự nhập, DFA có thể ở một trạng thái bất kỳ trong các trạng thái thuộc ε-closure(q_0) với q_0 là trạng thái bắt đầu của NFA. Gọi trạng thái này là T. Giả sử các trạng thái của T là các trạng thái đạt được từ q_0 trên các ký hiệu nhập và giả sử a là ký hiệu nhập kế tiếp. Khi đọc a, NFA có thể chuyển đến một trạng thái bất kỳ trong tập trạng thái δ(T, a). Khi chúng ta cho phép sự truyền rỗng, NFA có thể ở bất kỳ trạng thái nào trong ε-closure(δ(T, a)) sau khi đã đọc a.

Giải thuật :

Trạng thái bắt đầu ε-closure(q_0) chỉ là một trạng thái trong các trạng thái của DFA và trạng thái này chưa được đánh dấu;

While Có một trạng thái T của DFA chưa được đánh dấu **do**
Begin
Đánh dấu T; { xét trạng thái T}

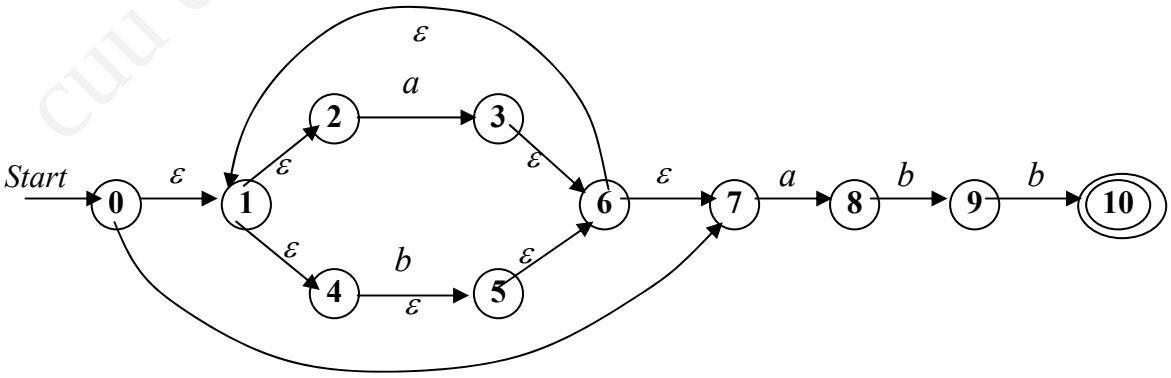
For Với mỗi ký hiệu nhập a ***do***
begin
U:= ε-closure(δ(T, a))
If U không có trong tập trạng thái của DFA ***then***
begin
Thêm U vào tập các trạng thái của DFA và trạng thái này chưa được đánh dấu;
δ[T, a] := U; {δ[T, a] là phần tử của bảng chuyển DFA}
end;
end;
End;

Ta xây dựng các trạng thái và bảng hàm chuyển cho DFA theo cách như sau :

- Mỗi trạng thái của DFA tượng trưng bởi một tập trạng thái của NFA mà NFA có thể chuyển đến sau khi đọc một chuỗi ký hiệu nhập gồm: tất cả sự truyền rỗng có thể xảy ra trước hoặc sau các ký hiệu được đọc.
- Trạng thái bắt đầu của DFA là ε-closure(q_0)
- Các trạng thái và hàm chuyển sẽ được thêm vào D bằng giải thuật trên.
- Một trạng thái của DFA là trạng thái kết thúc nếu nó là tập các trạng thái của NFA chứa ít nhất một trạng thái kết thúc của NFA.

Việc tính toán ε-closure(T) có thể xem như quá trình tìm kiếm một đồ thị của các nút từ các nút cho trước và đồ thị bao gồm toàn những cạnh có nhãn ε của NFA. Giải thuật đơn giản để tìm ε-closure(T) là dùng Stack để lưu giữ các trạng thái mà cạnh của chúng chưa được kiểm tra cho sự truyền rỗng.

Thí dụ 3.10 : Tạo DFA từ NFAε sau



Hình 3.6 – Thí dụ chuyển NFA có ε-dịch chuyển

Các bước xây dựng tập trạng thái cho DFA :

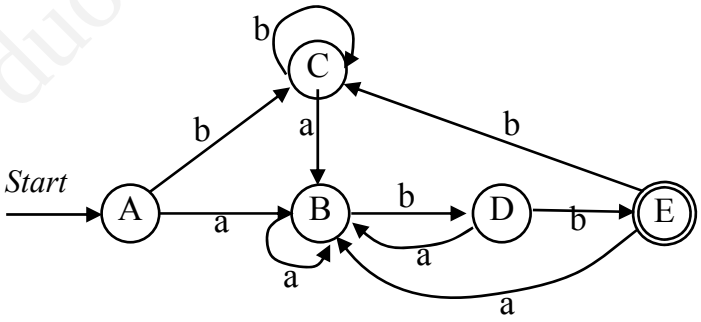
1) Trạng thái bắt đầu của DFA : ε-closure(0) = {0, 1, 2, 4, 7} = A*

2) ε-closure(δ(A, a)) = ε-closure({3, 8}) = {1, 2, 3, 4, 6, 7, 8} = B*
3) ε-closure(δ(A, b)) = ε-closure({5}) = {1, 2, 4, 5, 6, 7} = C*
4) ε-closure(δ(B, a)) = ε-closure({3, 8}) = B
5) ε-closure(δ(B, b)) = ε-closure({5, 9}) = {1, 2, 4, 5, 6, 7, 9} = D*
6) ε-closure(δ(C, a)) = ε-closure({3, 8}) = B
7) ε-closure(δ(C, b)) = ε-closure({5}) = C
8) ε-closure(δ(D, a)) = ε-closure({3, 8}) = B
9) ε-closure(δ(D, b)) = ε-closure({5, 10}) = {1, 2, 4, 5, 6, 7, 10} = E*
10) ε-closure(δ(E, a)) = ε-closure({3, 8}) = B
11) ε-closure(δ(E, b)) = ε-closure({5}) = C

Từ các tập trạng thái này, ta xác định được A là trạng thái bắt đầu, E là trạng thái kết thúc (vì trong E có chứa trạng thái 10 là trạng thái kết thúc của NFA) và bảng hàm chuyển của DFA như sau :

Trạng thái	**Ký hiệu nhập**	
	a	**b**
A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

Từ bảng hàm chuyển như trên, ta xây dựng sơ đồ chuyển trạng thái cho DFA tương đương nhận dạng cùng ngôn ngữ có dạng như sau :



Hình 3.7 – DFA tương đương cho thí dụ 3.10

Nhận xét : Mặc dù có sự khác nhau trong định nghĩa, ta thấy dạng không đơn định NFA được định nghĩa tổng quát hơn dạng đơn định DFA, nhưng rõ ràng khả năng nhận dạng cùng lớp ngôn ngữ của chúng là tương đương nhau. Trong thực tế, các máy tính số hoàn toàn là đơn định, trạng thái của chúng tại mỗi thời điểm là xác định được duy nhất từ một chuỗi nhập bất kỳ và trạng thái bắt đầu.

Câu hỏi :

i sao cần định nghĩa dạng không đơn định ?

Một số gợi ý câu trả lời:

1. Trong một số các bài toán mang tính chọn lựa, có nhiều hướng giải quyết (nhiều cách đi) như trong các chương trình trò chơi (games) thì thông thường hướng giải quyết tốt nhất (cách đi tốt nhất) là không biết trước được, nhưng có thể tìm thấy được bằng cách sử dụng chiến lược tìm kiếm quay lui (back-tracking). Khi có một vài khả năng chọn lựa có thể, ta chọn một khả năng trong chúng và đi theo hướng đó cho đến khi xác định hướng đó là tốt nhất hay chưa. Nếu chưa phải là hướng tốt nhất, ta phải quay về điểm quyết định cuối cùng trước đó và thử khảo sát theo một hướng khác. Một giải thuật mô phỏng quá trình tìm kiếm quay lui này là một giải thuật không đơn định.
2. Không đơn định đôi khi còn rất hữu hiệu trong việc giúp giải quyết các bài toán dễ dàng. Chẳng hạn, trong một số bài toán thì việc xây dựng một NFA có vẻ tự nhiên và đơn giản hơn việc tìm một DFA cho chúng. Tương tự như vậy, không đơn định còn là một cơ chế hiệu quả dùng mô tả văn phạm sinh ra ngôn ngữ một cách súc tích (sự chọn lựa các luật sinh sinh từ cùng một biến).
3. Trong thực tế, một vài kết quả là dễ dàng được chứng minh đối với NFA hơn là DFA. Vì vậy việc cho phép cơ chế không đơn định thường làm đơn giản hóa các lý luận hình thức mà không ảnh hưởng đến tính tổng quát của kết luận.

II. BIỂU THỨC CHÍNH QUY (RE : Regular Expressions)

Lớp ngôn ngữ được chấp nhận bởi một ôtômát hữu hạn cũng có thể được mô tả thông qua một dạng biểu thức ngắn gọn và súc tích gọi là biểu thức chính quy. Trong phần này, chúng ta sẽ giới thiệu sự kết hợp của các phép toán hợp, nối kết và bao đóng Kleene trên các tập hợp chuỗi để định nghĩa biểu thức chính quy và chứng tỏ rằng lớp ngôn ngữ được chấp nhận bởi một ôtômát hữu hạn thì thực sự là lớp ngôn ngữ được mô tả bởi biểu thức chính quy.

2.1. Định nghĩa

Cho Σ là một bộ chữ cái. Biểu thức chính quy trên Σ và các tập hợp mà chúng mô tả được định nghĩa một cách đệ quy như sau:

1) ∅ là biểu thức chính quy ký hiệu cho tập rỗng

2) ε là biểu thức chính quy ký hiệu cho tập {ε}

3) ∀a ∈ Σ, **a** là biểu thức chính quy ký hiệu cho tập {a}

4) Nếu r và s là các biểu thức chính quy ký hiệu cho các tập hợp R và S thì **(r + s)**, **(rs)** và **(r*)** là các biểu thức chính quy ký hiệu cho các tập hợp R ∪ S, RS, R^* tương ứng.

Trong khi viết biểu thức chính quy ta có thể bỏ bớt các dấu ngoặc đơn với lưu ý rằng thứ tự ưu tiên của các phép toán xếp theo thứ tự giảm dần là: **phép bao đóng, phép nối kết, phép hợp.**

Chẳng hạn : Biểu thức ((0(1*)) + 1) có thể viết là $01^*+ 1$.

Câu hỏi :

Như trên ta nói, biểu thức chính quy dùng ký hiệu cho một lớp ngôn ngữ. Bạn hãy thử liệt kê một vài chuỗi và hình dung lớp ngôn ngữ được ký hiệu bởi biểu thức chính quy $r = 01^*+ 1$ trên ?

Phép toán bao đóng dương cũng có thể được sử dụng khi viết biểu thức chính quy. Ta có thể viết rút gọn rr^* hay r^*r thành r^+.

Nếu cần thiết phân biệt thì ta sẽ dùng ký hiệu r cho biểu thức chính quy r và L(r) cho ngôn ngữ được ký hiệu bởi biểu thức chính quy r; ngược lại một cách tổng quát, ta có thể dùng r cho cả hai.

Thí dụ 3.11 : Một số biểu thức chính quy ký hiệu cho các ngôn ngữ :

. **00** là biểu thức chính quy biểu diễn tập {00}.

. **(0+1)*** ký hiệu cho tập hợp tất cả các chuỗi số 0 và số 1, kể cả chuỗi rỗng = {ε, 0, 1, 00, 01, 10, 11, 010, 011, 0010 ... }

. **(0+1)*00(0+1)*** ký hiệu cho tập hợp tất cả các chuỗi 0,1 có ít nhất hai số 0 liên tiếp.

= {00, 000, 100, 0000, 0001, 1000, 1001, 011001, ... }

. **(1+10)*** ký hiệu cho tất cả các chuỗi 0, 1 bắt đầu bằng số 1 và không có hai số 0 liên tiếp = {ε, 1, 10, 11, 1010, 111, 101010, ... }

. **(0+ε)(1+10)*** ký hiệu cho tất cả các chuỗi không có hai số 0 liên tiếp.

= {ε, 0, 01, 010, 1, 10, 01010, 0111, ... }

. **(0+1)*011** ký hiệu cho tất cả các chuỗi 0, 1 tận cùng bởi 011.

= {011, 0011, 1011, 00011, 11011, ... }

. **0*1*2*** ký hiệu cho tất cả các chuỗi có một số bất kỳ các số 0, theo sau là một số bất kỳ số 1 và sau nữa là một số bất kỳ số 2.

= {ε, 0, 1, 2, 01, 02, 12, 012, 0012, 0112, ... }

. **00*11*22*** ký hiệu cho tất cả các chuỗi trong tập $0^*1^*2^*$ với ít nhất một trong mỗi ký hiệu. $00^*11^*22^*$ có thể được viết gọn thành $0^+1^+2^+$

Thí dụ 3.12 : Biểu thức chính quy ký hiệu cho tập hợp các chuỗi tên biến đúng trong ngôn ngữ lập trình Pascal :

Một chuỗi tên biến (identifiers) được gọi là hợp lệ trong một chương trình Pascal nếu như nó bắt đầu bằng ít nhất một chữ cái và theo sau đó là các chữ cái, số, ký hiệu underline hoặc một vài ký hiệu cho phép khác trên bàn phím máy tính.

Biểu thức chính quy có dạng như sau :

$r = (A + \ldots + Z + a + \ldots + z)\ (A + \ldots + Z + a + \ldots + z + 0 + \ldots + 9 + \_ + \ldots)^*$

Thí dụ 3.13 : Biểu thức chính quy ký hiệu cho tập hợp các số nguyên trong ngôn ngữ lập trình Pascal :

Một chuỗi số nguyên trong một chương trình Pascal có thể bắt đầu bằng dấu âm (-) hoặc dấu dương (+) hay không chứa ký hiệu dấu, và theo sau đó là một chuỗi các ký hiệu số với ít nhất là một số.

Biểu thức chính quy có dạng như sau :

$$r = (\text{`+'} + \text{`-'} + \varepsilon)\ (0 + \ldots + 9\ (0 + \ldots + 9)^*$$

Nhận xét : Thông thường, việc tìm một biểu thức chính quy ký hiệu cho một ngôn ngữ khó hơn việc xác định ngôn ngữ được ký hiệu bởi một biểu thức chính quy vì không có giải thuật cho loại bài toán này.

2.2. Một số tính chất đại số của biểu thức chính quy

Dễ dàng chứng minh rằng, nếu cho r, s, t là các biểu thức chính quy thì ta có các đẳng thức sau :

1.	$r + s = s + r$	2.	$r + r = r$
3.	$r + (s+t) = (r+s) + t$	4.	$r\ (st) = (rs)\ t$
5.	$r\ (s+t) = rs + rt$	6.	$(r+s)\ t = rt + st$
7.	$r\varepsilon = \varepsilon\ r = r$	8.	$\varnothing r = r\varnothing = \varnothing$
9.	$r + \varnothing = r$	10.	$\varnothing^* = \varnothing$
11.	$(\varepsilon + r)^* = r^*$	12.	$r + r^* = r^*$
13.	$(r^*)^* = r^*$	14.	$(r^* s^*)^* = (r+s)^*$

Trong đó, ta có r = s có nghĩa là L(r) = L(s).

III. SỰ TƯƠNG ĐƯƠNG GIỮA ÔTÔMÁT HỮU HẠN VÀ BIỂU THỨC CHÍNH QUY

Như trên đã nói, các ngôn ngữ được chấp nhận bởi ôtômát hữu hạn cũng là các ngôn ngữ được mô tả bởi biểu thức chính quy. Chính vì sự tương đương này, mà người ta gọi ngôn ngữ chấp nhận bởi ôtômát hữu hạn là các tập chính quy. Trong phần này, thông qua hai định lý, ta sẽ chỉ ra bằng quy nạp theo kích thước của (số phép toán trong) biểu thức chính quy rằng có tồn tại một NFA với ε-dịch chuyển chấp nhận cùng ngôn ngữ; đồng thời với mỗi DFA cũng có một biểu thức chính quy xác định chính ngôn ngữ của nó.

ĐỊNH LÝ 3.3: Nếu r là biểu thức chính quy thì tồn tại một NFA với ε-dịch chuyển chấp nhận L(r).

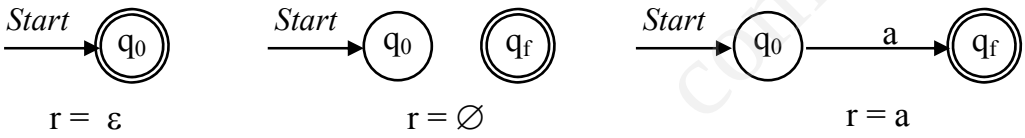
Chứng minh

Ta sẽ chứng minh quy nạp theo số phép toán của biểu thức chính quy r rằng có tồn tại một NFA M với ε-dịch chuyển có một trạng thái kết thúc và không có các phép chuyển khỏi trạng thái này chấp nhận biểu thức chính quy r: L(M) = L(r).

. r không có phép toán:

Vậy r phải là ∅, ε hoặc a (với a ∈ Σ).

Các NFA dưới đây thoả mãn điều kiện:



Hình 3.7 - Các NFAε cho các kết hợp đơn

. r có chứa các phép toán:

Giả sử định lý đúng với r có ít hơn i phép toán, i ≥ 1.

Xét r có i phép toán. Có 3 trường hợp :

1) $r = r_1 + r_2$.

Cả hai biểu thức chính quy r_1, r_2 có ít hơn i phép toán, vậy ta có 2 ôtômát hữu hạn NFA M_1 (Q_1, Σ_1, δ_1, q_1, $\{f_1\}$) và M_2 (Q_2, Σ_2, δ_2, q_2, $\{f_2\}$) sao cho $L(M_1) = L(r_1)$ và $L(M_2) = L(r_2)$. Vì các trạng thái có thể thay đổi tên nên ta giả sử hai tập trạng thái Q_1 và Q_2 là rời nhau. Đặt q_0 là trạng thái bắt đầu mới và $\{f_0\}$ là tập trạng thái kết thúc mới, ta xây dựng NFA M ($Q_1 \cup Q_2 \cup \{q_0, f_0\}$, $\Sigma_1 \cup \Sigma_2$, δ, q_0, $\{f_0\}$), trong đó δ được xác định như sau:

. $\delta(q_0, \varepsilon) = \{q_1, q_2\}$

. $\delta(q, a) = \delta_1(q, a)$ với $q \in Q_1 - \{f_1\}$ và $a \in \Sigma_1 \cup \{\varepsilon\}$

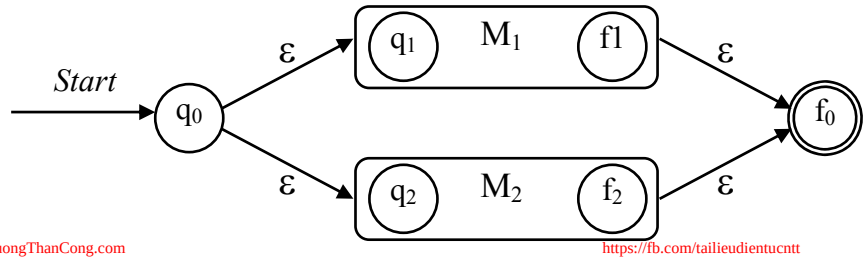
. $\delta(q, a) = \delta_2(q, a)$ với $q \in Q_2 - \{f_2\}$ và $a \in \Sigma_2 \cup \{\varepsilon\}$

. $\delta(f_1, \varepsilon) = \delta(f_2, \varepsilon) = \{f_0\}$

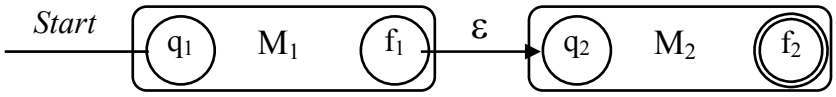
Chú ý do giả thiết quy nạp là không có phép chuyển nào ra khỏi f_1, f_2 trong M_1, M_2. Vì vậy tất cả các phép chuyển của M_1 và M_2 đều có trong M. Cách xây dựng M chỉ ra trong hình a. Bất kỳ đường đi nào trong sơ đồ chuyển của M từ q_0 tới f_0 phải bắt đầu bằng cách đi tới q_1 hoặc q_2 bằng nhãn ε. Nếu đường đi qua q_1 thì nó theo một đường đi nào đó trong M_1 tới f_1 rồi sau đó tới f_0 bằng nhãn ε.

Tương tự trong trường hợp đường đi qua q_2. Có một đường đi từ q_0 đến f_0 nhãn x khi và chỉ khi có đường đi nhãn x trong M_1 từ q_1 đến f_1 hoặc có đường đi nhãn x trong M_2 từ q_2 đến f_2.

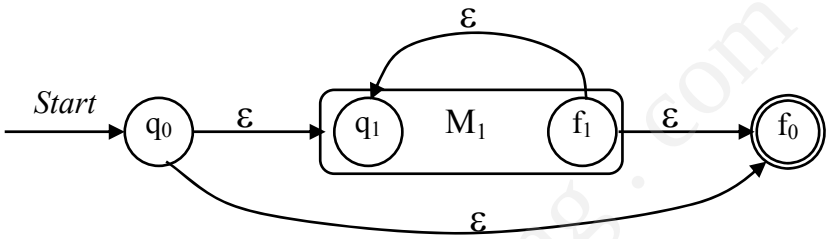
Vậy $L(M) = L(M_1) \cup L(M_2)$



Hình a - Phép hợp



Hình b - *Phép nối kết*



Hình c - Phép bao đóng

Hình 3.8 - Các NFAε cho kết hợp phức

2) $r = r_1 r_2$

Đặt M_1 và M_2 là các ôtômát NFA như trong trường hợp trên và ta xây dựng ôtômát M $(Q, \Sigma, \delta, \{q_1\}, \{f_2\})$, trong đó δ được xác định như sau:

. $\delta(q, a) = \delta_1(q, a)$ với $q \in Q_1 - \{f_1\}$ và $a \in \Sigma_1 \cup \{\varepsilon\}$

. $\delta(f_1, \varepsilon) = \{q_2\}$

. $\delta(q, a) = \delta_2(q, a)$ với $q \in Q_2$ và $a \in \Sigma_2 \cup \{\varepsilon\}$

Cách xây dựng M chỉ ra trong hình b. Mỗi đường đi trong M từ q_1 tới f_2 là đường đi có nhãn x từ q_1 tới f_1 sau đó là một cung từ f_1 tới q_2 nhãn ε và tiếp đến là đường đi từ q_2 tới f_2.

Vậy $L(M) = \{xy / x \in L(M_1)$ và $y \in L(M_2)\}$ hay $L(M) = L(M_1) L(M_2)$.

3) $r = r^*$

Đặt M_1 $(Q_1, \Sigma_1, \delta_1, q_1, \{f_1\})$ và $L(M_1) = r_1$.

Xây dựng M $(Q_1 \cup \{q_0, f_0\}$ Σ_1, δ, q_0, $\{f_0\})$, trong đó δ được cho:

. $\delta(q_0, \varepsilon) = \delta(f_1, \varepsilon) = \{q_1, f_0\}$

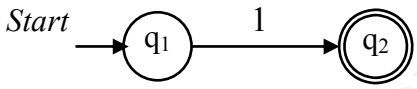
. $\delta(q, a) = \delta_1(q, a)$ với $q \in Q_1 - \{f_1\}$ và $a \in \Sigma_1 \cup \{\varepsilon\}$

Cách xây dựng M được chỉ ra trong hình c. Mỗi đường đi từ q_0 tới f_0 gồm: hoặc đường đi từ q_0 tới f_0 bằng nhãn ε; hoặc đường đi từ q_0 tới q_1 bằng nhãn ε và sau đó là đường đi từ q_1 tới f_1 trên chuỗi thuộc L(M), rồi đến f_0 bằng nhãn ε. Như vậy có đường đi từ q_0 tới f_0 nhãn là x nếu và chỉ nếu ta có thể viết $x = x_1 x_2 \ldots x_j$ với $j \geq 0$ (trường hợp $j = 0$ khi $x = \varepsilon$) $\forall x_i \in L(M_1)$. Vậy $L(M) = L(M_1)^*$.

Thí dụ 3.14 : Xây dựng NFAε chấp nhận lớp ngôn ngữ được ký hiệu bởi biểu thức chính quy $r = 01^* + 1$.

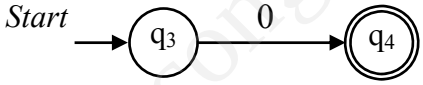
Ta thấy L(r) = { 1, 0, 01, 011, 0111, 01111, 011111, … } là tập ngôn ngữ chứa các bit đơn 0, 1 và các chuỗi bit nhị phân bắt đầu bằng bit 0, theo sau là một chuỗi bit 1 với độ dài tuỳ ý.
Theo quy luật thứ tự ưu tiên, biểu thức $01^* + 1$ thực chất là $(0(1^*)) + 1$, vì vậy nó có dạng $\mathbf{r_1} + \mathbf{r_2}$ với $\mathbf{r_1} = 01^*$ và $\mathbf{r_2} = 1$.
Ta sẽ lần lượt xây dựng các NFA cho các biểu thức chính quy con, sau đó dựa vào các quy tắc kết hợp để xây dựng NFA cho toàn bộ biểu thức chính quy đã cho.

. NFA cho $r_2 = 1$ dễ dàng để xây dựng :

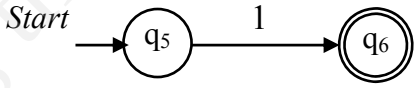


. NFA cho $r_1 = \mathbf{01^*}$:
Ta tách $r_1 = r_3\, r_4$, trong đó $r_3 = 0$ và $r_4 = 1^*$

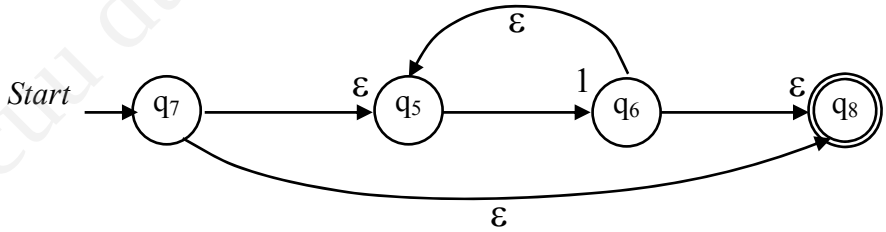
+ NFA cho $r_3 = \mathbf{0}$:



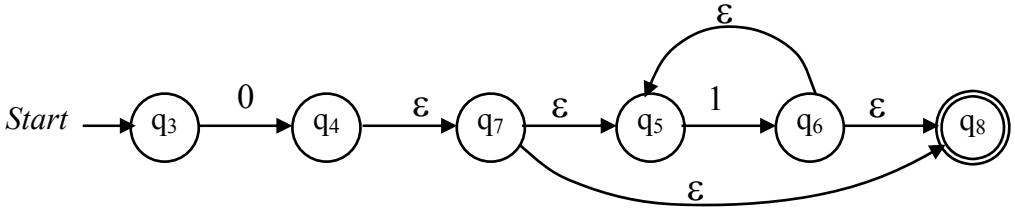
+ NFA $r_4 = \mathbf{1^*}$:
Ta viết $r_4 = r_5^*$, trong đó $r_5 = 1$.
NFA cho $r_5 = \mathbf{1}$:



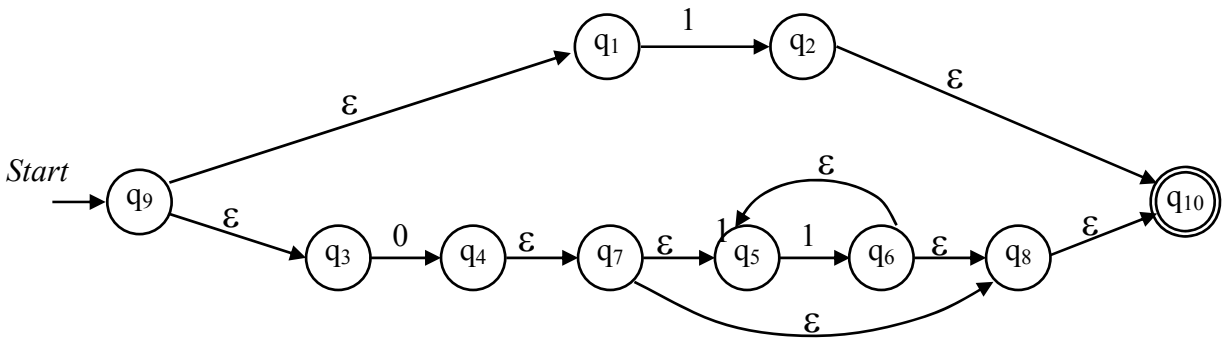
Theo quy tắc 3) ta xây dựng được NFA cho $r_4 = r_5^* = 1^*$ như sau :



Theo quy tắc 2) ta xây dựng được NFA cho $r_1 = r_3\, r_4 = 01^*$ như sau :



Cuối cùng, theo quy tắc 1) ta xây dựng NFA cho $r = r_1 + r_2 = 01^* + 1$ như sau :



Hình 3.9 - NFAε cho ví dụ 3.13

Phần chứng minh của Định lý 3.3 trên cũng chính là cơ sở của giải thuật chuyển đổi một biểu thức chính quy thành ôtômát hữu hạn. Một điểm cần lưu ý là thứ tự ưu tiên của các phép toán được sử dụng trong biểu thức chính quy, điều này rất quan trọng cho quá trình tách biểu thức chính quy thành các biểu thức con trong những trường hợp viết biểu thức chính quy ở dạng tắt (không có dấu ngoặc).

Bây giờ, ta cần chứng tỏ rằng mọi tập hợp được chấp nhận bởi một ôtômát hữu hạn thì cũng được ký hiệu bởi một số biểu thức chính quy.

<u>ĐỊNH LÝ 3.4</u> : Nếu L được chấp nhận bởi một DFA, thì L được ký hiệu bởi một biểu thức chính quy.
Chứng minh

Đặt L là tập hợp được chấp nhận bởi DFA M ($\{q_1, q_2,..., q_n\}$, Σ, δ, q_1, F).
Đặt R^k_{ij} là tập hợp tất cả các chuỗi x sao cho $\delta(q_i, x) = q_j$ và nếu $\delta(q_i, y) = q_l$, với y là tiền tố bất kỳ của x, khác x hoặc ε, thì $l \le k$. Tức là R^k_{ij} là tập hợp tất cả các chuỗi làm cho ôtômát đi từ trạng thái qi tới qj không đi ngang qua trạng thái nào (được đánh số) lớn hơn k. (Chú ý, khái niệm "*đi ngang qua một trạng thái*" có nghĩa là có phép chuyển vào và ra khỏi trạng thái đó, nên i hoặc j đều có thể lớn hơn k). Vì chỉ có n trạng thái nên R^n_{ij} sẽ là tập hợp tất cả các chuỗi làm ôtômát đi từ qi tới qj.

Ta định nghĩa R^k_{ij} một cách đệ quy như sau:

$$\mathbf{R^k_{ij}} = R^{k-1}_{ik} (R^{k-1}_{kk})^* R^{k-1}_{kj} \cup R^{k-1}_{ij} \qquad \textbf{(1)}$$

$$R^0_{ij} = \begin{cases} \{ a \mid \delta(q_i, a) = q_j \} & \text{nếu } i \neq j \\ \{ a \mid \delta(q_i, a) = q_j \} \cup \{\varepsilon\} & \text{nếu } i = j \end{cases}$$

Một cách hình thức, R^k_{ij} định nghĩa như trên là các chuỗi nhập hay nguyên nhân đưa M từ q_i tới q_j không đi ngang qua trạng thái cao hơn q_k, nghĩa là xảy ra hoặc một trong hai trường hợp sau :

1) Nằm trong R^{k-1}_{ij} (để không bao giờ đi ngang qua một trạng thái nào cao như q_k).
2) Bao gồm một chuỗi trong R^{k-1}_{ik} (chuỗi làm M chuyển đến q_k), theo sau bởi không hoặc nhiều chuỗi trong R^{k-1}_{kk} (chuỗi làm M chuyển từ q_k trở về q_k mà không ngang qua q_k hoặc một trạng thái nào cao hơn) và cuối cùng là một chuỗi trong R^{k-1}_{kj} (chuỗi làm M chuyển từ q_k đến q_j).

Ta sẽ chỉ ra rằng với mỗi i, j và k tồn tại biểu thức chính quy r^k_{ij} ký hiệu cho ngôn ngữ R^k_{ij}. Ta quy nạp theo k như sau:

. k = 0 : khi đó R^0_{ij} là tập hợp hữu hạn các chuỗi có một ký hiệu hoặc ε. Vậy r^0_{ij} có thể viết dưới dạng $a_1 + a_2 + ... + a_p$ (hoặc $a_1 + a_2 + ... + a_p + \varepsilon$ nếu i = j). Trong đó $\{a_1, a_2,..., a_p\}$ là tập hợp tất cả các ký hiệu a sao cho $\delta(q_i, a) = q_j$. Nếu không có ký hiệu a nào như thế thì ∅ (hoặc ε khi i = j) ký hiệu cho r^0_{ij}.

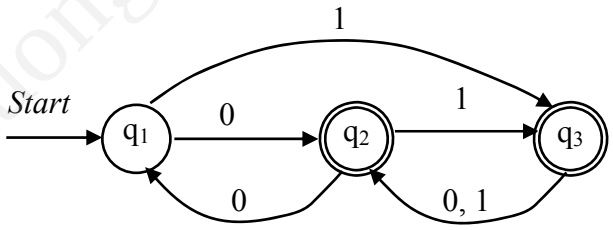
. Công thức (1) cho R^k_{ij} chỉ liên quan đến các phép toán trên biểu thức chính quy: hợp, nối kết, và bao đóng. Hơn nữa theo giả thiết quy nạp, với mỗi l, k và m tồn tại biểu thức chính quy r^{k-1}_{lm} sao cho $L(r^{k-1}_{lm}) = R^{k-1}_{lm}$. Vậy đối với r^k_{ij} ta có thể chọn biểu thức chính quy :

$$(r^{k-1}_{ik})\,(r^{k-1}_{kk})^*\,(r^{k-1}_{kj}) + r^{k-1}_{ij}$$

Cuối cùng ta có nhận xét rằng $L(M) = \cup_{q_j \in F} R^n_{1j}$ vì R^n_{1j} ký hiệu cho tất cả các nhãn của tất cả các đường đi từ q_1 tới q_j.

Vậy L(M) được ký hiệu bởi biểu thức chính quy $r = r^n_{1j1} + r^n_{1j2} + ... + r^n_{1jp}$, trong đó tập $F = \{q_{j1}, q_{j2},..., q_{jp}\}$

Thí dụ 3.15 : Viết biểu thức chính quy ký hiệu cho ngôn ngữ được chấp nhận bởi DFA sau :



Hình 3.10 – DFA cho ví dụ 3.13

Gọi DFA được chỉ ra trong hình 3.10 là M ($\{q_1, q_2, q_3\}$, $\{0, 1\}$, δ, q_1, $\{q_2, q_3\}$). Ta thấy, tập hợp tất cả các chuỗi được chấp nhận bởi DFA trên là các chuỗi làm cho ôtômát chuyển từ trạng thái bắt đầu q_1 đến một trong hai trạng thái kết thúc q_2 và q_3 và không chuyển qua số tối đa là 3 (k = 3) trạng thái của ôtômát. Vậy ta cần viết biểu thức chính quy ký hiệu cho tập hợp này như sau :

$$r = r^3_{12} + r^3_{13}$$

Theo công thức đã được chứng minh trong Định lý, ta có thể tính được các giá trị r^k_{ij} với i, j là chỉ số các trạng thái từ 1 đến 3 và với k = 0, 1 và 2, như chỉ ra trong bảng sau:

	k = 0	*k = 1*	*k = 2*
r^k_{11}	**ε**	**ε**	**(00)***

r^k_{12}	**0**	**0**	$\mathbf{0(00)^*}$
r^k_{13}	**1**	**1**	$\mathbf{0^*1}$
r^k_{21}	**0**	**0**	$\mathbf{0(00)^*}$
r^k_{22}	**ε**	**ε + 00**	$\mathbf{(00)^*}$
r^k_{23}	**1**	**1 + 01**	$\mathbf{0^*1}$
r^k_{31}	∅	∅	$\mathbf{(0 + 1)(00)^*0}$
r^k_{32}	**0 + 1**	**0 + 1**	$\mathbf{(0 + 1)(00)^*}$
r^k_{33}	**ε**	**ε**	$\mathbf{ε + (0 + 1)0^*1}$

Bằng cách dùng các công thức tương đương như (r + s) t = rt + st và $(ε + r)^* = r^*$ để đơn giản các biểu thức, chẳng hạn khi tính biểu thức :

$$r^1_{22} = r^0_{21}(r^0_{11})^* r^0_{12} + r^0_{22} = \mathbf{0(ε)^*0 + ε} = \mathbf{00 + ε}$$

Tương tự, khi đơn giản biểu thức

$$r^2_{13} = r^1_{12}(r^1_{22})^* r^1_{23} + r^1_{13} = \mathbf{0(00 + ε)^*(1 + 01) + 1}$$

ta nhận thấy $\mathbf{(00 + ε)^*}$ tương đương với $(00)^*$ và **(1 + 01)** thì tương đương với **(ε + 0)1** nên ta rút gọn :

$$r^2_{13} = \mathbf{0(00)^*(ε + 0)1 + 1}$$

Mặt khác, chú ý rằng $\mathbf{(00)^*(ε + 0)}$ thì tương đương với $\mathbf{0^*}$, vì thế $\mathbf{0(00)^*(ε + 0)1 + 1}$ cũng bằng $\mathbf{00^*1 + 1}$ hay cuối cùng là $\mathbf{0^*1}$.

Để hoàn thành việc xây dựng biểu thức chính quy cho M, ta cần tính r^3_{12} và r^3_{13}. Ta viết:

$$\begin{aligned} r^3_{12} &= r^2_{13}(r^2_{33})^* r^2_{32} + r^2_{12} \\ &= \mathbf{0^*1(ε + (0 + 1)0^*1)^*(0 + 1)(00)^* + 0(00)^*} \\ &= \mathbf{0^*1((0 + 1)0^*1)^*(0 + 1)(00)^* + 0(00)^*} \end{aligned}$$

và

$$\begin{aligned} r^3_{13} &= r^2_{13}(r^2_{33})^* r^2_{33} + r^2_{13} \\ &= 0^*1(ε + (0 + 1)0^*1)^*(ε + (0 + 1))0^*1) + 0^*1 \\ &= \mathbf{0^*1((0 + 1)0^*1)^*} \end{aligned}$$

Vậy biểu thức chính quy có dạng :

$$r = r^3_{12} + r^3_{13} = \mathbf{0^*1((0 + 1)0^*1)^*(ε + (0 + 1)(00)^*) + 0(00)^*}$$

Iv. MỘT VÀI ỨNG DỤNG CỦA ÔTÔMÁT HỮU HẠN

Có nhiều kiểu phần mềm thiết kế nhằm đặc tả sự chuyển đổi tự động từ dạng biểu thức chính quy sang việc cài đặt máy tính một cách hiệu quả tương ứng với ôtômát hữu hạn. Trong phần này, ta sẽ đề cập đến hai ứng dụng trong số chúng.

4.1. Bộ phân tích từ vựng

Các ký hiệu từ vựng (token) trong một ngôn ngữ lập trình thì hầu hết không có sự ngoại lệ, được biểu diễn như các tập hợp chính quy. Chẳng hạn, các định danh của ALGOL: các chữ cái viết hoa hoặc thường, theo sau bởi một dãy bất kỳ của chữ cái (letter) hoặc chữ số (digit) với độ dài không giới hạn có thể được biểu diễn như sau :

$$(\text{letter})\ (\text{letter} + \text{digit})^*$$

trong đó "letter" thay thế cho A + B +...+ Z + a + b +...+ z và "digit" là 0 + 1 +...+ 9.

Một ví dụ khác, các định danh của FORTRAN có độ dài giới hạn là 6 và các chữ cái chỉ cho phép dùng chữ viết hoa hoặc ký hiệu $ được biểu diễn như sau :

$$(\text{letter})\ (\varepsilon + \text{letter} + \text{digit})^5$$

với "letter" là $ + A + B + ... + Z .
Trong SNOBOL, các hằng số số học có thể được biểu diễn như sau :

$$(\varepsilon + -)\ (\text{digit}^+\ (\bullet\ \text{digit}^* + \varepsilon) + \bullet\ \text{digit}^+)$$

Một số công cụ phát sinh bộ phân tích từ vựng nhận input như một dãy các biểu thức chính quy mô tả các ký hiệu từ vựng và phát sinh một ôtômát hữu hạn đơn giản nhận dạng mọi ký hiệu từ vựng. Thông thường, chúng chuyển đổi biểu thức chính quy thành một NFA với ε-dịch chuyển và sau đó xây dựng tập hợp con các trạng thái để có thể phát sinh DFA một cách trực tiếp hơn là tìm cách loại bỏ các phép chuyển nhãn ε. Mỗi trạng thái kết thúc xác định ký hiệu từ vựng cụ thể đã tìm thấy. Hàm chuyển của FA sẽ được mã hóa bằng một trong vài cách nhằm chiếm ít không gian hơn so với bảng hàm chuyển tổ chức dưới dạng mảng hai chiều. Bộ phân tích từ vựng được thiết lập bằng cách phát sinh một chương trình cố định thông dịch các bảng mã, cùng với các bảng minh họa cụ thể sự nhận dạng của FA trên các ký hiệu từ vựng (viết dưới dạng các biểu thức chính quy). Bộ phân tích từ vựng dạng này có thể được dùng như một chương trình con độc lập (module) trong một trình biên dịch ngôn ngữ.

4.2. Trình soạn thảo văn bản

Hiển nhiên, các trình soạn thảo văn bản hoặc các chương trình tương tự cho phép thay thế một chuỗi bởi mọi chuỗi kết hợp với một biểu thức chính quy cho trước.

Chẳng hạn, trình soạn thảo văn bản ***ed*** trong UNIX cho phép một câu lệnh như sau :
/aba*c/
để tìm sự xuất hiện đầu tiên của chuỗi có dạng như trên. Hay câu lệnh :

s/bbb*/b/

cho phép thay thế các chuỗi có dạng **bbb*** thành chuỗi có một ký tự **b**.

Hay trong các câu lệnh của MS-DOS và NC, ví dụ câu lệnh :
Del *tmp*.???
sẽ cho phép xóa đi tất cả các file với tên tập tin bắt đầu bằng ***tmp***, sau đó là một chuỗi bất kỳ và có phần mở rộng là 3 ký tự tùy ý. Dấu * trong trường hợp này ký hiệu cho một chuỗi bất kỳ, còn dấu **?** ký hiệu cho một ký tự tùy ý. Đây cũng là một dạng ký hiệu của biểu thức chính quy thay thế cho chuỗi.

Hay chẳng hạn, một ví dụ về xử lý chuỗi khác được áp dụng cho việc tìm kiếm theo mẫu trên các trang Web.

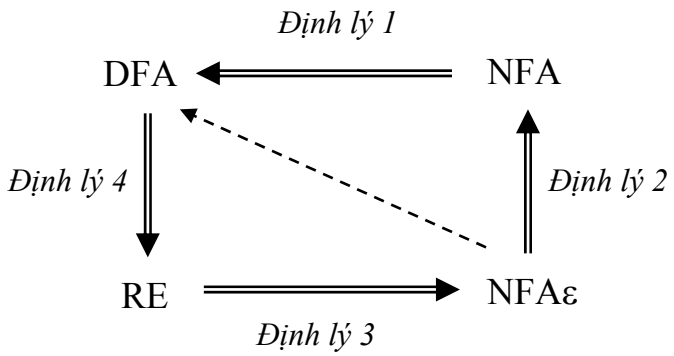
Trong tất cả các ví dụ trên, ký hiệu * xác định "mọi" biểu thức $a_1 + a_2 + ... + a_n$ trong đó các a_i là tất cả các ký tự cho phép trong máy tính trừ ký tự xuống dòng (newline). Ta có thể chuyển một biểu thức chính quy r sang DFA chấp nhận mọi r. Chú ý rằng sự hiện diện của ký hiệu * sẽ cho phép ta nhận dạng một thành phần của L(r) bắt đầu từ bất kỳ vị trí nào trong dòng. Để làm được điều này, các ứng dụng phải thực hiện quá trình chuyển đổi từ một biểu thức chính quy sang NFA. Và vì cơ chế hoạt động của NFA khá phức tạp nên thông thường ngay sau đó, NFA lại phải được biến đổi tiếp thành dạng DFA tương đương. Tuy nhiên, sự chuyển đổi từ một biểu thức chính quy sang DFA tốn nhiều thời gian hơn việc sử dụng DFA để kiểm tra các mẫu bằng cách duyệt qua chúng một lần, tuy DFA có thể có số trạng thái là hàm mũ của độ dài biểu thức chính quy.

Thực tế, trong trình soạn thảo văn bản UNIX, biểu thức chính quy với mọi r được chuyển thành một NFA có ε-dịch chuyển và sau đó NFA này được mô phỏng một cách trực tiếp.

Câu hỏi :

Hãy liên hệ một số các ứng dụng thực tế khác dùng cơ chế ôtômát hữu hạn ?

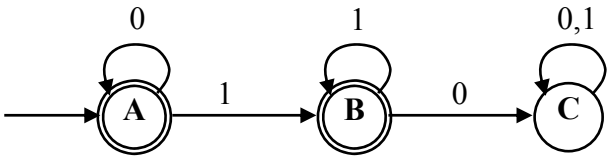
Tổng kết chương III: Phần nội dung chương III tập trung nghiên cứu cơ chế hoạt động của các dạng ôtômát hữu hạn, mối tương quan giữa chúng, cũng như sự tương đương của chúng với biểu thức chính quy. Tùy theo các yêu cầu thực tế của ứng dụng, ta có thể chuyển từ dạng phức tạp nhất sang các dạng đơn giản hơn. Có thể tóm tắt sự tương quan giữa các Định lý trong chương này theo sơ đồ sau :



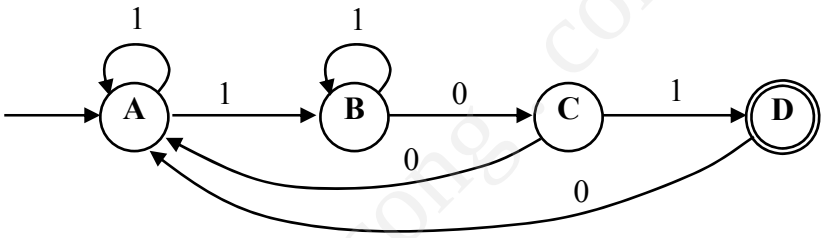
BÀI TẬP CHƯƠNG III

3.1. Mô tả ngôn ngữ được chấp nhận bởi các ôtômát hữu hạn với sơ đồ chuyển được cho như sau :

a)



b)



3.2. Xây dựng các sơ đồ chuyển ôtômát hữu hạn chấp nhận các ngôn ngữ sau trên bộ chữ cái Σ = {0, 1}

a) Tập các chuỗi kết thúc là 00.
b) Tập các chuỗi có 3 ký hiệu 0 liên tiếp.
c) Tập các chuỗi mà ký hiệu thứ 3 kể từ cận phải của chuỗi là 1.
d) Tập mọi chuỗi mà bất cứ chuỗi con nào có độ dài bằng 5 đều có chứa ít nhất 2 con số 0.

3.3. Tìm các sơ đồ chuyển ôtômát hữu hạn đoán nhận các ngôn ngữ sau :

a) Tập các chuỗi trên {0, 1} có chứa một số chẵn các số 0 và một số lẻ các số 1
b) Tập các chuỗi trên {0, 1} có độ dài chia hết cho 3.
c) Tập các chuỗi trên {0, 1} không chứa 101 như một chuỗi con.

3.4. Xây dựng DFA tương đương với mỗi NFA sau :

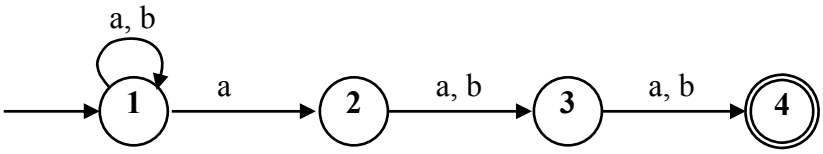
a) $N_1(\{0,1,2,3\},\{a,b\},\delta_1,0,\{3\})$ với δ_1

δ_1	a	b
0	{0, 1}	{1}
1	{2}	{2}
2	{3}	∅
3	{3}	{3}

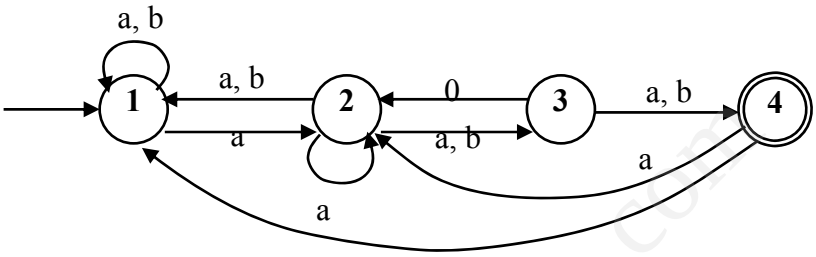
b) $N_2(\{0,1,2,3\}, \{a,b\}, \delta_2,0, \{1,3\})$ với δ_2

δ_2	a	b
0	{1, 3}	{1}
1	{2}	{1, 2}
2	{3}	{0}
3	∅	{0}

c)

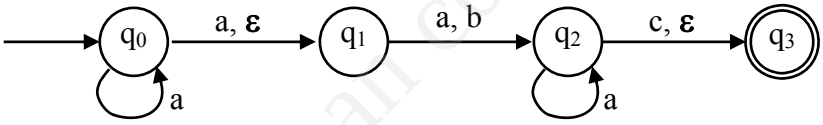


d)

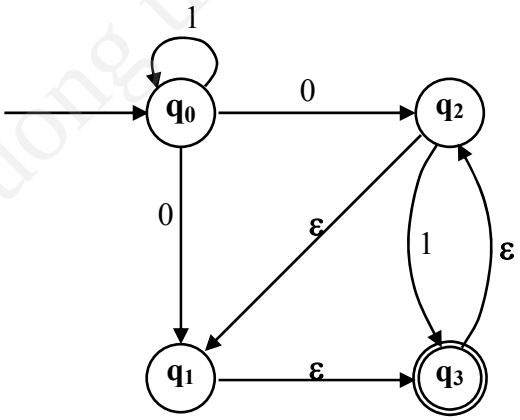


3.5. Tìm NFA không có ε-dịch chuyển nhận dạng cùng ngôn ngữ với các NFA sau :

a)



b)



3.6. Viết biểu thức chính quy và vẽ NFA đoán nhận các ngôn ngữ sau :

a) Tập hợp các chuỗi trên Σ = {1, 2, 3} sao cho ký hiệu cuối cùng đã có xuất hiện trước đó .

b) Tập hợp các chuỗi trên Σ = {0, 1} trong đó có một cặp ký tự 0 cách nhau bởi một chuỗi con có độ dài 4i, với i ≥ 0 nào đó.

3.7. Viết biểu thức chính quy cho mỗi ngôn ngữ sau trên Σ = { 0, 1} :

a) Tập hợp các chuỗi trong đó mọi cặp 0 liên tiếp đều xuất hiện trước mọi cặp 1 liên tiếp.

b) Tập hợp các chuỗi chứa nhiều nhất một cặp 0 liên tiếp và nhiều nhất một cặp 1 liên tiếp.

3.8. Mô tả (bằng lời) ngôn ngữ được các biểu thức chính quy sau đặc tả :

a) $0(0 + 1)^* 0$
b) $(0+ 1)^*0(0 + 1) (0 + 1)$
c) $(11+ 0)^*(00+ 1)$
d) $(1+ 01+ 001)^*(\varepsilon + 0 + 00)$
e) $[00 + 11 + (01+ 10) (00+ 11)^*(01+ 10)]^*$

3.9. Chứng tỏ các biểu thức chính quy sau ký hiệu cho cùng một ngôn ngữ :

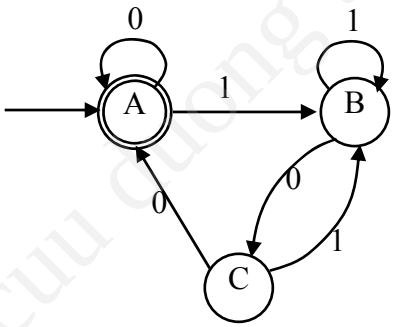
$(aa)^*$, $(aa)^* + (\varnothing a)$, $(aa + aaaa)^*$, $(aa)^* (aa)^*$

3.10. Vẽ NFA với ε-dịch chuyển được cho bởi các biểu thức chính qui sau. Sau đó, hãy chuyển sang DFA tương đương :

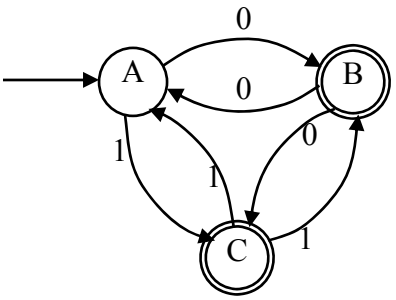
a) $(a^* + b^*)^*$
b) $((\varepsilon + a) b^*)^*$
c) $(a + b)^* abb (a + b)^*$
d) $ab + (a + bb) a^*b$
e) $(a + ab + aab)^*(\varepsilon+ a+ aa)$
f) $10 + (0 + 11)0^*1$
g) $01 [((10)^*+ 111)^* + 0]^*1$

3.11. Hãy tìm các biểu thức chính qui tương ứng với các sơ đồ chuyển trạng thái sau:

a)



b)



BÀI TẬP LẬP TRÌNH

3.12. Viết chương trình trong Pascal / C mô phỏng một FA chấp nhận ngôn ngữ được biểu diễn bởi biểu thức chính quy sau :

$$[A \dots Z, a \dots Z]^+ [A \dots Z, a \dots Z, 0 \dots 9, \_]^* + [0 \dots 9]^+ + \text{ op}$$

3.13. Viết chương trình cho ra một FA tương ứng khi đầu vào là một biểu thức chính quy.

3.14. Viết chương trình cho ra DFA khi đầu vào là một NFA.