Bài 3.
PHÂN TÍCH TỪ VỰNG

Hoàng Anh Việt
Viện CNTT&TT - ĐHBKHN

Kiểm tra bài trước

- Bài tập 2.1:

  Cho văn phạm phi ngữ cảnh: $\mathbf{S \rightarrow S\ S + \mid S\ S * \mid a}$

  $\Rightarrow$Xây dựng cây PTCP cho câu nhập: $\mathbf{aa+a^*}$

- Bài 2.2 Đâu là văn phạm mơ hồ:

  a) $S \rightarrow 0\ S\ 1 \mid 0\ 1$

  b) $S \rightarrow +\ S\ S \mid -\ S\ S \mid a$

  c) $S \rightarrow S\ (\ S\ )\ S \mid \in$

  d) $S \rightarrow a\ S\ b\ S \mid b\ S\ a\ S \mid \in$

  e) $S \rightarrow a \mid S + S \mid S\ S \mid S\ * \mid (\ S\ )$

Mục đích

- Sau khi học xong chương này, sinh viên sẽ nắm được:
  - Các kỹ thuật xác định và cài đặt bộ PTTV.
  - Xây dựng các lược đồ cho các biểu thức chính quy mô tả ngôn ngữ.
  - DFA và NFA. Các automata hữu hạn xác định và không xác định dùng để nhận dạng chính xác ngôn ngữ.
  - Sử dụng công cụ có sẵn Lex để sinh ra bộ PTTV

Điều kiện

- Kiến thức cần có:
  - Kiến thức cơ bản về NFA và DFA
  - Cách chuyển đổi giữa các Automata.

Tài liệu tham khảo

[1] Slide bài giảng

[2] Compilers : Principles, Technique and Tools - Alfred V.Aho, Jeffrey D.Ullman - Addison - Wesley Publishing Company, 1986.

[3] Automata and Formal Language, An Introduction- Dean Kelley- Prentice Hall, Englewood Cliffs, New Jersey 07632

[4] Compilers course, CS 143 summer 2010, Standford University.

Nội dung

1. Vai trò của bộ phân tích từ vựng
2. Lữu trữ tạm chương trình nguồn
3. Đặc tả Token
4. Nhận dạng Token
5. Sơ đồ dịch
6. Automat hữu hạn
7. Từ biểu thức chính quy đến NFA
8. Tổng kết quá trình PTTV
9. Thiết kế bộ sinh bộ PTTV

Nội dung

1. Vai trò của bộ phân tích từ vựng

2. Lữu trữ tạm chương trình nguồn

3. Đặc tả Token

4. Nhận dạng Token

5. Sơ đồ dịch

6. Automat hữu hạn

7. Từ biểu thức chính quy đến NFA

8. Tổng kết quá trình PTTV

9. Thiết kế bộ sinh bộ PTTV

1. Vai trò của bộ phân tích từ vựng

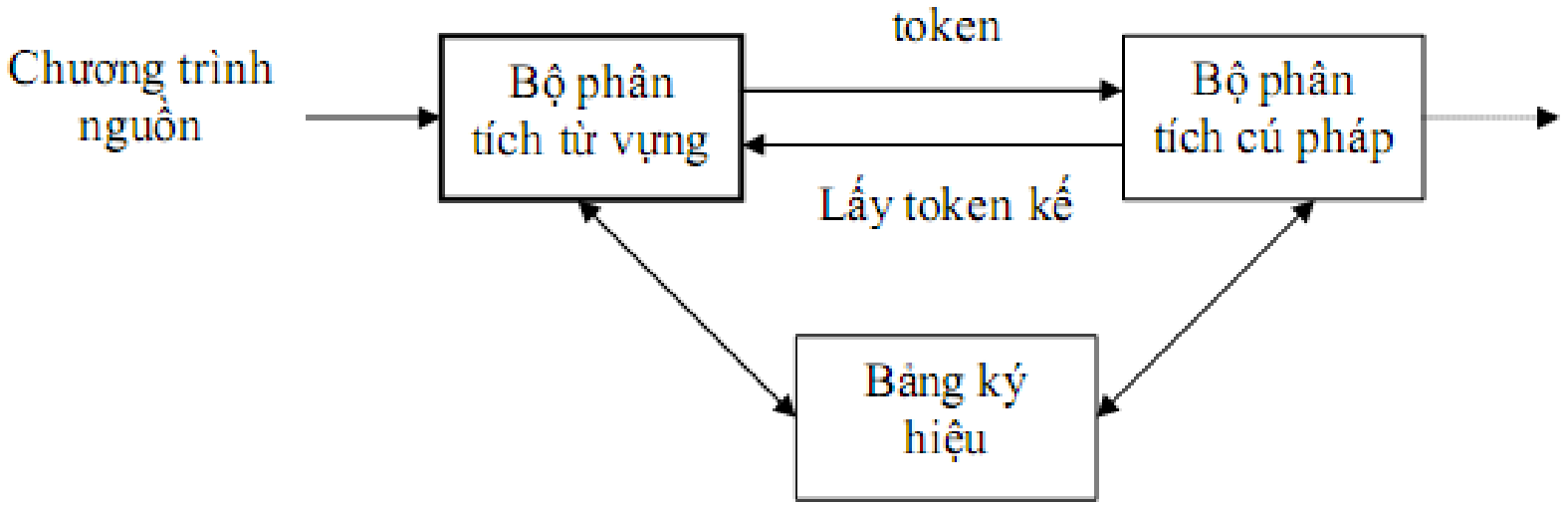
1.1 Ý nghĩa của giai đoạn PTTV

1.2 Các khái niệm

1.3 Thuộc tính của Token

1.4 Lỗi từ vựng

1. Vai trò của bộ phân tích từ vựng



Hình 3.1 - *Giao diện của bộ phân tích từ vựng*

1.1 Ý nghĩa của giai đoạn PTTV

- Làm cho việc thiết kế đơn giản và dễ hiểu hơn
- Hiệu quả của trình biên dịch được cải thiện nhờ một số chương trình xử lý chuyên dụng.
- Tính đa tương tích của trình biên dịch được cải thiện.

1.2 Các khái niệm

- Từ tố (Token): là các ký hiệu kết thúc trong văn phạm đối với ngôn ngữ nguồn. Ví dụ: từ khóa, toán tử, dấu câu, hằng, định danh…
- Trị từ vựng (Lexeme) của một token là một chuỗi ký tự biểu diễn cho token đó
- Mẫu từ vựng (pattern) là qui luật mô tả một tập các trị từ vựng kết hợp với một token nào đó.

1.2 Các khái niệm

Token	Trị từ vựng minh họa	Mô tả của mẫu từ vựng
const	const	const
if	if	if
relation	<, <=, =, < >, >, >=	< hoặc <= hoặc = hoặc <> hoặc > hoặc >=
id	pi, count, d2	Mở đầu là chữ cái theo sau là chữ cái, chữ số
num	3.1416, 0, 5	Bất kỳ hằng số nào
literal	" hello "	Mọi chữ cái nằm giữa " và " ngoại trừ "

Hình 3.2 - *Các ví dụ về token*

1.3 Thuộc tính của Token

- Khi có nhiều mẫu từ vựng khớp với trị từ vựng, bộ PTTV phải cung cấp thêm thông tin và cất chúng vào bảng danh biểu (Ví dụ trị từ vựng).

➢ Token luôn mang trong mình một thuộc tính duy nhất là con trỏ để chỉ đến vị trí của nó trong bảng danh biểu.

➢ Khi một token được chuyển đến bộ phân tích cú pháp nó sẽ có dạng.

<Token, thuộc tính>

1.3 Thuộc tính của Token

- Ví dụ 3.1: Câu lệnh: X=Y*2, được viết như một dãy các bộ:
  - <**id**, con trỏ trong bảng ký hiệu của X>
  - <**assign_op,** >
  - <**id**, con trỏ trong bảng ký hiệu của Y>
  - <**mult_op**, >
  - <**num**, giá trị nguyên 2>

Chú ý: Một số bộ không cần giá trị thuộc tính, thành *phần đầu tiên là đủ nhận dạng trị từ vựng*

1.4 Lỗi từ vựng

- Chỉ một số ít lỗi được phát hiện tại bước PTTV.
- Ví dụ: **fi (i>=m)…**

  ⇒**fi** lỗi viết sai từ khóa **if ?**

  ⇒**Hay** một id (danh biểu) chưa được khai báo?
- Các chiến lược khắc phục lỗi:
  - *Xóa đi 1 ký tự dư*
  - *Xen thêm 1 ký tự bị mất*
  - *Thay thế ký tự sai bằng ký tự đúng*
  - *Chuyển đổi hai ký tự kề nhau*

Nội dung

1. Vai trò của bộ phân tích từ vựng
2. **Lữu trữ tạm chương trình nguồn**
3. Đặc tả Token
4. Nhận dạng Token
5. Sơ đồ dịch
6. Automat hữu hạn
7. Từ biểu thức chính quy đến NFA
8. Thiết kế bộ sinh bộ PTTV

2. Lưu trữ tạm chương trình nguồn

2.1 Cặp bộ đệm

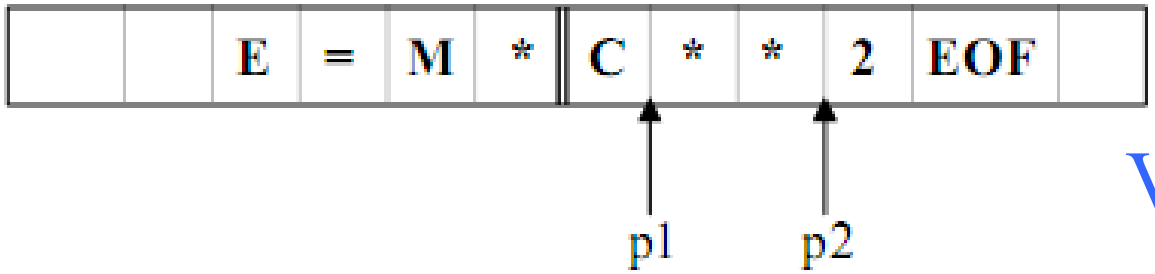
2.2 Khóa canh

2. Lưu trữ tạm chương trình nguồn

- Vấn đề: Đọc từng ký tự chương trình nguồn tốn nhiều thời gian và ảnh hưởng tốc độ dịch
- Giải quyết: Đọc một lúc một chuỗi ký tự và lưu vào bộ đệm buffer.
- Thế nào cho trọn vẹn Token?

2.1 Cặp bộ đệm

- Vùng đệm chia làm 2 nửa với kích thước N (1024 hoặc 4096)
- Sử dụng 2 con trỏ P1, P2 để dò tìm.
  - P1 đặt tại vị trí đầu của 1 trị từ vựng
  - P2 dịch chuyển để xác định trị từ vựng cho token.



Vấn đề?

Hình 3.3 - *Cặp hai nửa vùng đệm*

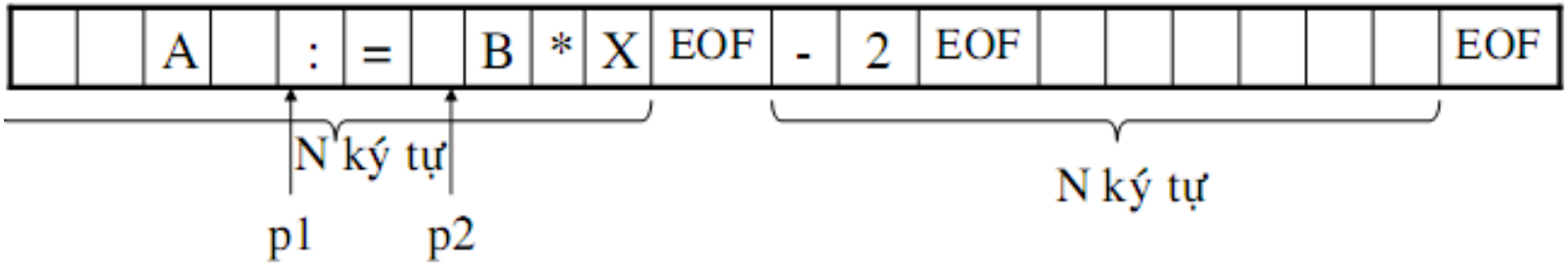
2.1 Cặp bộ đệm

Giải thuật:

```
if p2 ở ranh giới một nửa bộ đệm then
   begin
        lấp đầy N ký hiệu nhập mới vào nửa bên phải.
        p2 := p2 + 1;
   end
else if p2 ở tận cùng bên phải bộ đệm then
   begin
        lấp đầy N kỳ hiệu nhập vào nửa bên trái bộ đệm.
        chuyển p2 về ký tự tận cùng bên trái của bộ đệm.
   end
else p2 := p2 + 1;
```

2.2 Khóa canh

- Chỉ đọc N-1 ký tự vào mỗi nửa buffer.
- Ký tự N là eof.



Hình 3.4- *Khóa canh oef tại cuối mỗi vùng đệm*

2.2 Khóa canh

Giải thuật:

p2 := p2 + 1;

If p2 ^ eof **then**

if p2 ở ranh giới một nửa bộ đệm **then**

begin

chất đầy N kỳ hiệu nhập vào nửa bên phải bộ đệm ;

p2 := p2 + 1

end

2.2 Khóa canh

else if p2 ở tận cùng bên phải bộ đệm **then**

begin

lấp đầy N ký hiệu vào nửa bên trái bộ đệm;

chuyển p2 về đầu bộ đệm

end

else /* dừng sự phân tích từ vựng*/

end

Nội dung

1. Vai trò của bộ phân tích từ vựng
2. Lữu trữ tạm chương trình nguồn
3. **Đặc tả Token**
4. Nhận dạng Token
5. Sơ đồ dịch
6. Automat hữu hạn
7. Từ biểu thức chính quy đến NFA
8. Thiết kế bộ sinh bộ PTTV

3. Đặc tả token

3.1 Chuỗi và ngôn ngữ

3.2 Các phép toán trên ngôn ngữ

3.3 Biểu thức chính quy

3.4 Các tính chất đại số của biểu thức chính quy

3.5 Định nghĩa chính quy

3.6 Ký hiệu viết tắt.

3.1 Chuỗi và ngôn ngữ

- **Chuỗi:**
  - Tập hợp hữu hạn các ký tự
  - Độ dài chuỗi là số ký tự trong chuỗi
  - Chuỗi rỗng **ε** là chuỗi có độ dại 0.
- **Ngôn ngữ:**
  - Là tập hợp các chuỗi
  - Có thể chỉ bao gồm 1 chuỗi rỗng ký hiệu là **Ø**

3.2 Các phép toán trên ngôn ngữ

- Xét 2 ngôn ngữ L và M:
  - **Hợp:** của L và M là **L ∪ M** = {s|s ∈ L hoặc s ∈ M}
  - **Ghép:** của L và M là: **LM**= {st| s ∈L và t ∈ M}
  - **Bao đóng Kleen** của L: L^* = {ghép của 0 hoặc nhiều L}
  - **Bao đóng dương** của L={ghép của 1 hoặc nhiều L}

3.2 Các phép toán trên ngôn ngữ

- **Ví dụ 3.2**

  L={A,B,….Z,a,b,…z}

  D={0,1,…9}

- **L ∪ D** là tập hợp các chữ cái và số.
- **LD** là tập hợp các chuỗi bao gồm một chữ cái và một chữ số.
- $\mathbf{L^4}$ là tập hợp tất cả các chuỗi có 4 chữ cái
- $\mathbf{L^*}$ là tập hợp các chuỗi của các chữ cái và rỗng.
- $\mathbf{L(L \cup D)^*}$ là tập hợp tất cả các chuỗi mở đầu bằng 1 chữ cái và theo sau là chữ cái hoặc số.
- $\mathbf{D^+}$ là tập hợp các chuỗi gồm 1 hoặc nhiều chữ số.

3.3 Biểu thức chính quy (regular Expression)

- **Nhắc lại:** Trong NNLT, 1 biến (danh biểu) là một phần tử của tập hợp $\mathbf{L(L \cup D)^*}$

  => có thể viết: **biến=letter(letter|digit)***

Đây biểu thức chính quy!

3.3 Biểu thức chính quy (regular Expression)

- Biểu thức chính quy:
  - Được xây dựng trên một tập hợp các luật xác định.
  - Mỗi BTCQ r đặc tả cho một ngôn ngữ L(r).
  - 2 BTCQ là tương đương nếu cùng đặc tả một tập hợp

Các luật xác định BTCQ trên tập Alphabet $\sum$:

1. ε là một biểu thức chính quy đặc tả 1 chuỗi rỗng {ε}
2. Nếu a $\in \sum$ thì a là BTCQ r đặc tả tập hợp các chuỗi {a}
3. r và s là các BTCQ đặc tả các ngôn ngữ L(r) và L(s)
   1. (r)|(s) là một btcq đặc tả L(r) ∪ L(s)
   2. (r)(s) là 1 btcq đặc tả L(r)L(s).
   3. $(r)^*$ là 1 btcq đặc tả $(L(r))^*$

3.3 Biểu thức chính quy (regular Expression)

- **Ví dụ 3.3** Cho $\sum$ = { a, b}
  - BTCQ **a|b** đặc tả {a,b}
  - BTCQ **(a|b)(a|b)** đặc tả tập hợp {aa,ab,ba,bb}
  - BTCQ $\mathbf{a}^*$ đặc tả { ε, a, aa, aaa, ... }
  - BTCQ **(a | b)*** đặc tả {a, b, aa,bb, ...}. Tập này có thể đặc tả bởi (a*b*)*.
  - BTCQ **a | a* b** đặc tả {a, b, ab, aab,... }

3.4 Các tính chất đại số của BTCQ

Tính chất	Mô tả
r | s = s | r	| có tính chất giao hoán
r | (s | t) = (r | s) | t	| có tính chất kết hợp
(rs) t = r (st)	Phép ghép có tính chất kết hợp
r (s | t) = rs | rt (s | t) r = sr | tr	Phép ghép phân phối đối với phép |
εr = r rε = r	ε là phần tử đơn vị của phép ghép
r* = (r | ε)*	Quan hệ giữa r và ε
r* * = r *	* có hiệu lực như nhau

Hình 3.5 - *Một số tính chất đại số của biểu thức chính quy*

3.5 Định nghĩa chính quy

- Định nghĩa chính quy là chuỗi định nghĩa có dạng:

$$d_1 \rightarrow r_1$$

……

$$d_n \rightarrow r_n$$

Trong đó: $\mathbf{d_i}$ là 1 tên còn $\mathbf{r_i}$ là 1 BTCQ

- **Ví dụ 3.4:** Tập hợp các danh biểu trong Pascal

letter → A | B | ...|Z | a| b |...| z
digit → 0 |1| ...| 9
id → letter (letter | digit)*

3.5 Định nghĩa chính quy

- **Ví dụ 3.5:**Các số không dấu trong Pascal là các chuỗi 5280, 39.37, 6.336E4 hoặc 1.894E-4. Định nghĩa chính quy sau đặc tả tập các số này là :

digit → 0 | 1 |...| 9

digits → digit digit*

optional_fraction → . digits | ε

optional_exponent → (E (+ | - | ε) digits) | ε

num → digits optional_fraction optional_exponent

3.6 Ký hiệu viết tắt

1. Một hoặc nhiều: dùng dấu +
2. Không hoặc một: dùng dấu ?
   - Ví dụ 3.6: r | ε được viết tắt là r ?
   - Ví dụ 3.7: Viết tắt cho định nghĩa chính quy tập hợp số num trong ví dụ 3.5

digit → 0 | 1 |... | 9

digits → digit $^{+}$

optional_fraction → (. digits) ?

optional_exponent → (E (+ | -) ? digits) ?

num → digits optional_fraction optional_exponent

3. Lớp ký tự

3.6 Ký hiệu viết tắt

[abc] = a | b | c

[a - z] = a | b |... | z

Sử dụng lớp ký hiệu chúng ta có thể mô tả danh biểu như là một chuỗi sinh ra bởi biểu thức chính quy :

[A - Z a - z] [A - Z a - z 0 – 9] *

Nội dung

1. Vai trò của bộ phân tích từ vựng
2. Lữu trữ tạm chương trình nguồn
3. Đặc tả Token
4. **Nhận dạng Token**
5. Sơ đồ dịch
6. Automat hữu hạn
7. Từ biểu thức chính quy đến NFA
8. Tổng kết quá trình PTTV
9. Thiết kế bộ sinh bộ PTTV

4. Nhận dạng Token

- Cho văn phạm G:

stmt -> **if** exp **then** stmt

| **if** exp **then** stmt **else** stmt

| €

exp -> term **relop** term |term

term -> **id** |**num**

- Các ký hiệu kết thúc: if, then, else, relop, id, num được cho bởi **định nghĩa chính quy**.

4. Nhận dạng Token

if → if

then → then

else → else

relop → < | <= | = | <> | > | >=

id → **letter (letter | digit)** *

num → **digit** $^{+}$ **(. digit** $^{+}$**)** ? **(E (+ | -)** ? **digit** $^{+}$**)** ?

- Định nghĩa chính quy khoảng trắng

delim → **blank | tab | newline**

ws → **delim**$^{+}$

4. Nhận dạng Token

Biểu thức chính quy	Token	Trị thuộc tính
ws	-	-
If	if	-
Then	Then	-
Else	Else	-
Id	Id	Con trỏ trong bảng ký hiệu
Num	Num	Giá trị số
<	Relop	LT (Less than)
<=	Relop	LE (Less or Equal)
=	Relop	EQ (Equal)
<>	Relop	NE (not equal)
>	Relop	GT (Greater than)
>=	Relop	GE (Greater or Equal)

Hình 3.6: *Mẫu biểu thức chính quy cho 1 số token*

Nội dung

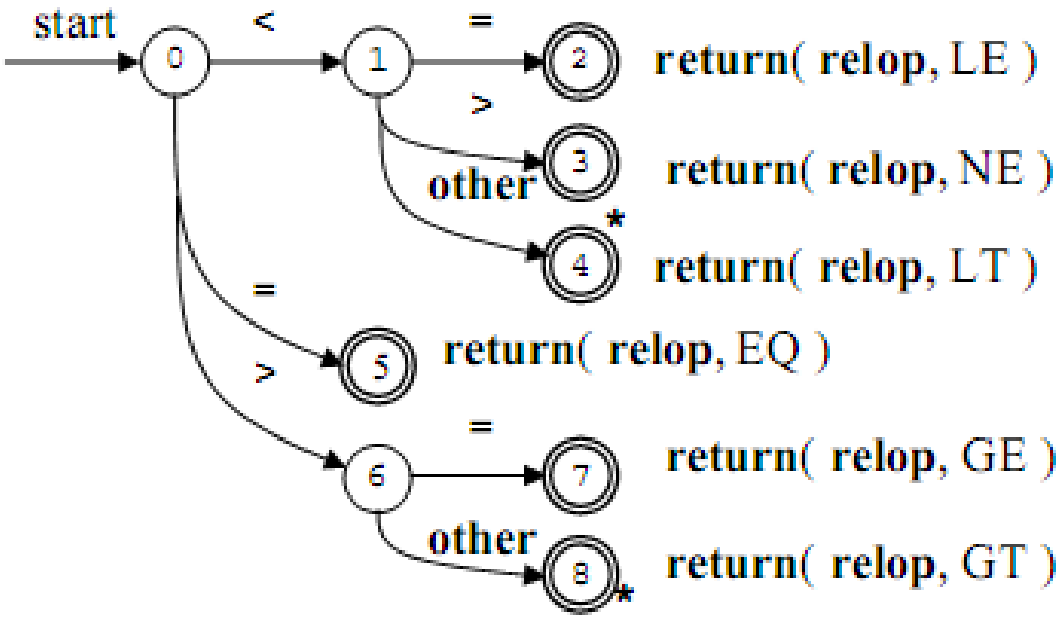
1. Vai trò của bộ phân tích từ vựng
2. Lữu trữ tạm chương trình nguồn
3. Đặc tả Token
4. Nhận dạng Token
5. **Sơ đồ dịch**
6. Automat hữu hạn
7. Từ biểu thức chính quy đến NFA
8. Thiết kế bộ sinh bộ PTTV

5. Sơ đồ dịch

- Để dễ nhận dạng Token
- Mỗi nhóm Token có một sơ đồ dịch
- Nếu xảy ra thất bại khi đang theo một SDD thì lui con trỏ lại vị trí đầu và kích hoạt SDD tiếp theo.
- Nếu thất bại trong mọi SDD=> lỗi từ vựng và cần khởi động thủ tục khắc phục
- Mỗi SDD gồm: trạng thái, các cạnh nối.

5. Sơ đồ dịch

- Sơ đồ dịch nhận dạng cho Token **relop**:

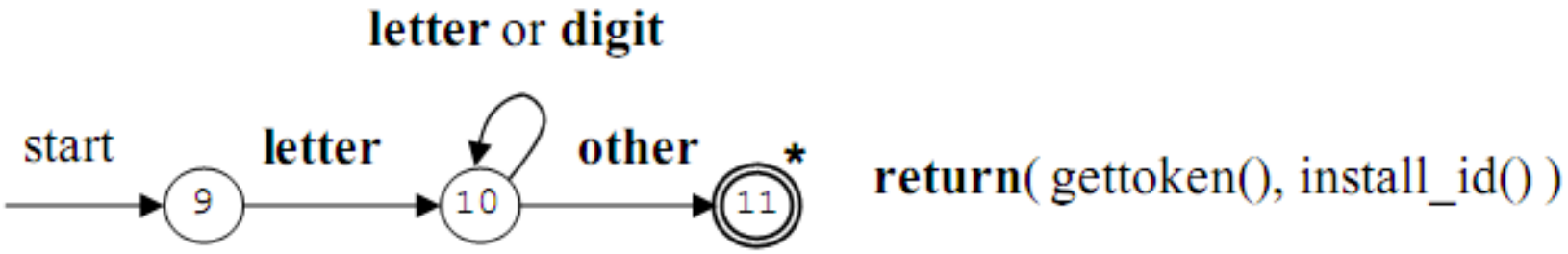


Hình 3.7 - *Sơ đồ dịch cho các toán tử quan hệ*

Chú ý: ký tự * chỉ trạng thái đọc quá 1 ký tự, cần quay lui con trỏ

5. Sơ đồ dịch

- Sơ đồ dịch nhận dạng Token **Id**

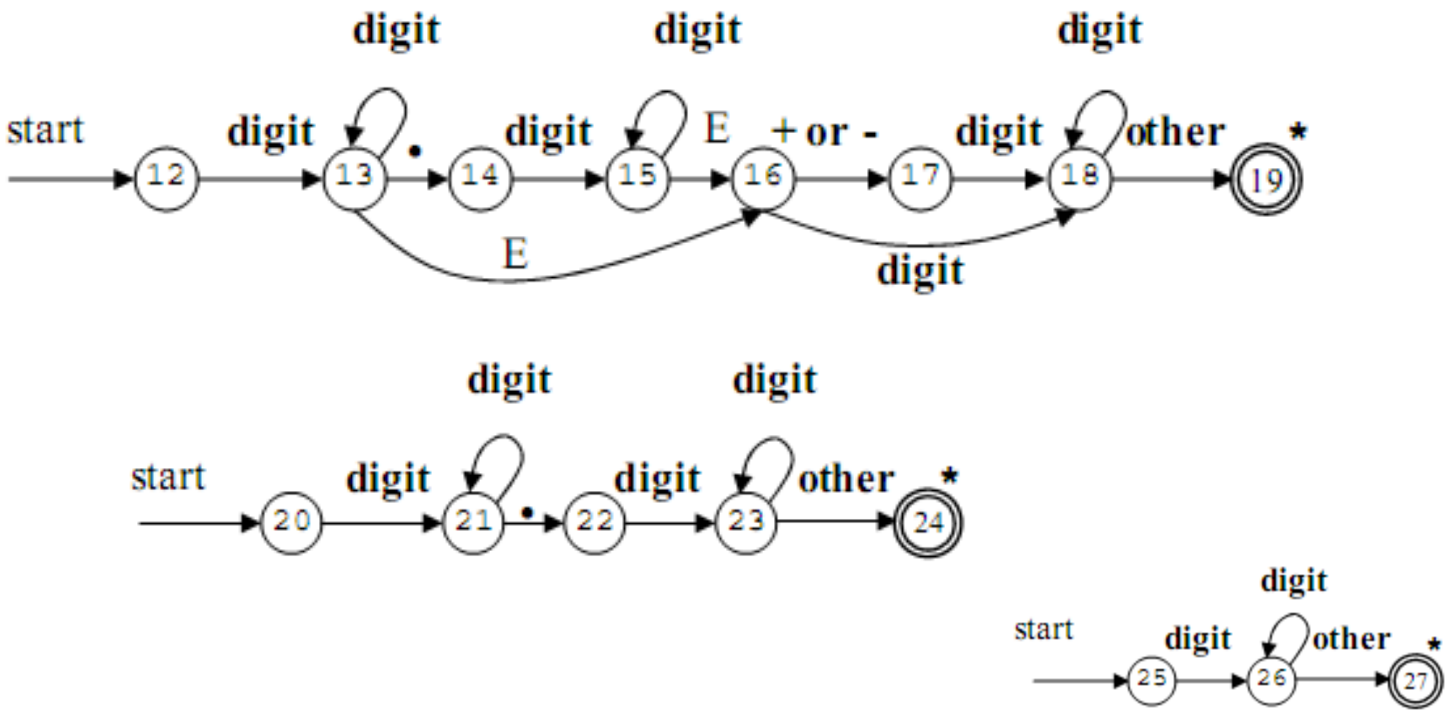


Hình 3.8 - *Sơ đồ dịch cho các danh biểu và từ khóa*

- gettoken( ) và install_id( ) tương ứng để nhận token và các thuộc tính trả về.

5. Sơ đồ dịch

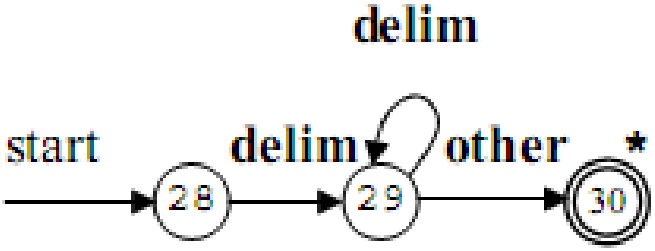
- Sơ đồ dịch nhận dạng **num**



Hình 3.9 - *Sơ đồ dịch cho các số không dấu trong Pascal*

5. Sơ đồ dịch

- Sơ đồ dịch nhận dạng khoảng trắng **ws**



Hình 3.10 - *Sơ đồ dịch cho các khoảng trắng*

Có gì đặc biệt?

Nội dung

1. Vai trò của bộ phân tích từ vựng
2. Lữu trữ tạm chương trình nguồn
3. Đặc tả Token
4. Nhận dạng Token
5. Sơ đồ dịch
6. **Automat hữu hạn**
7. Từ biểu thức chính quy đến NFA
8. Thiết kế bộ sinh bộ PTTV

6. Automat hữu hạn

- Kiến thức nền:
  - Lý thuyết Văn phạm
  - Lý thuyết ngôn ngữ
  - Lý thuyết tập hợp
  - Các kỹ thuật chứng minh: Quy nạp, phản chứng.
- Tài liệu tham khảo:
  1. Bài giảng lý thuyết ngôn ngữ hình thức và Automat- Hồ Văn Quân [2002]
  2. An Introduction to Formal Languages and Automat- Peter Linz [1990]

6. Automat hữu hạn

6.1 Các khái niệm Automat

6.2 Phân loại và ứng dụng

6.3 Automat hữu hạn đơn định (DFA)

6.4 Automat hữu hạn không đơn định (NFA)

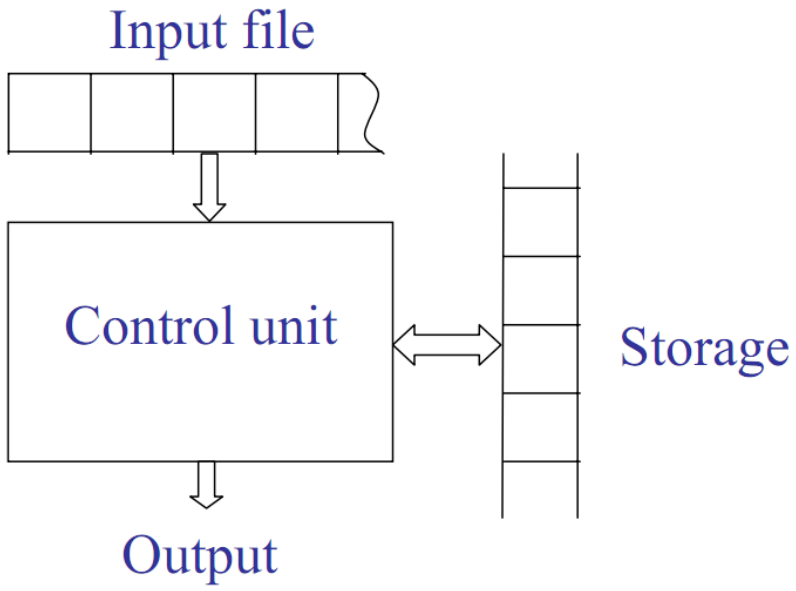
6.5 Chuyển từ NFA sang DFA

6.1 Các khái niệm Automat

- Automat là gì?
  - Dịch là *máy tự động*, là thiết bị có thể thực hiện công việc mà không cần sự can thiệp của con người.
  - Nó hoạt động dựa trên một số quy tắc và dựa vào các quy tắc này con người lập trình cho nó hoạt động theo ý muốn của mình.
  - Máy tính số ngày nay là một máy tư động điển hình và mạnh nhất.

6.1 Các khái niệm Automat

- Là một mô hình trừu tượng của máy tính số bao gồm các thành phần chủ yếu sau:



6.1 Các khái niệm Automat

- **Thiết bị đầu vào** (input file): là nơi mà chuỗi nhập (input string) được ghi lên, và được automat đọc nhưng không thay đổi được nội dung. Nó được chia thành các ô, mỗi ô giữ một ký hiệu.
- **Cơ cấu nhập** (input mechanism): là bộ phận có thể đọc input file từ trái sang phải, một ký tự tại một thời điểm. Nó cũng có thể dò tìm được điểm kết thúc của chuỗi nhập (oef, #).

6.1 Các khái niệm Automat

- **Bộ nhớ tạm** (temporary storage): là thiết bị gồm một số không giới hạn ô nhớ (cell), mỗi ô có thể giữ 1 ký hiệu từ một bảng chữ cái (không cần giống bảng chữ cái nhập). Automat có thể đọc và thay đổi nội dung của các ô nhớ này.
- **Đơn vị điều khiển** (Control Unit): mỗi automat có một đơn vị điều khiển, cái mà có thể ở trong 1 trạng thái bất kỳ trong một số hữu hạn các trạng thái nội, và có thể chuyển đổi trạng thái trong một kiểu được định nghĩa sẵn có nào đó.

6.1 Các khái niệm Automat

- Hoạt động của Automat:
  - Tại một thời điểm bất kỳ đã cho, đơn vị điều khiển đang ở trong một **trạng thái nội** (internal state) nào đó, và cơ cấu nhập là **đang quét** (scanning) một ký hiệu cụ thể nào đó trên **Input file.**
  - **Trạng thái nội** của đơn vị điều khiển tại thời điểm kế tiếp được xác định bởi trạng thái kế (next state) hay bởi hàm chuyển trạng thái (transition function).
  - Trong suốt quá trình chuyển trạng thái từ khoảng thời gian này đến khoảng thời gian kế, **kết quả** (output) có thể được sinh ra và thông tin trong **bộ nhớ lưu trữ** có thể được **thay đổi.**

6.1 Các khái niệm Automat

- Một số định nghĩa khác:
  - Trạng thái nội: là một trạng thái của đơn vị điều khiển mà nó có thể đạt được.
  - Trạng thái kế: là một trạng thái nội của đơn vị điều khiển mà nó sẽ đạt tới ở thời điểm kế tiếp.
  - Hàm chuyển trạng thái: là hàm gởi ra trạng thái kế của automat dựa trên ***trạng thái hiện hành***, ***ký hiệu nhập hiện hành*** được quét, và ***thông tin hiện hành trong bộ nhớ tạm***.

6.1 Các khái niệm Automat

- Một số định nghĩa khác:
  - Cấu hình (configuration): được sử dụng để tham khảo đến bộ thông tin:
    - » Trạng thái cụ thể mà đơn bị điều khiển đang có
    - » Vị trí của cơ cấu nhập trên thiết bị nhập.
    - » Nội dung hiện hành của bộ nhớ tạm
  - Di chuyển (move): là sự chuyển trạng thái của automat từ một cấu hình này sang cấu hình kế tiếp.

6.2 Phân loại và ứng dụng

- Dựa vào hoạt động của Automat, có đơn định hay không:
  - Automat đơn định (Deterministic Automat): là automat trong đó mỗi di chuyển (move) được xác định duy nhất bởi cấu hình hiện tại. Sự duy nhất này thể hiện tính đơn định.
  - Automat không đơn định (non-deterministic automat): là automat mà tại mỗi thời điểm có một vài khả năng lựa chọn để di chuyển. Việc có một vài khả năng lựa chọn thể hiện tính không đơn định.

6.2 Phân loại và ứng dụng

- Dựa vào kết quả xuất ra của automat:
  - Accepter: là automat mà đáp ứng ở ngõ ra của nó được giới hạn trong hai trạng thái đơn giả “yest” hay “no”. “yes” tương ứng với việc chấp nhận chuỗi nhập, “no” tương ứng với việc từ chối, không chấp nhận chuỗi nhập.
  - Transducer: là automat tổng quát hơn, có khả năng sinh ra các chuỗi ký tự ở đầu ra. Máy tính số là một transducer điển hình.

6.2 Phân loại và ứng dụng

- Một vài ứng dụng:
  - Cung cấp kiến thức nền tảng cho việc xây dựng các ngôn ngữ lập trình, chương trình dịch.
  - Ứng dụng vào các lĩnh vực xử lý chuỗi:
    - Các chức năng tìm kiếm, thay thế trong các trình soạn thảo
    - Sửa lỗi chính tả, chú thích từ loại…
  - Ứng dụng vào lĩnh vực thiết kế số.
  - …

6.3 Automat hữu hạn đơn định (DFA)

- Định nghĩa:
  - Một Automat hữu hạn đơn định (deterministic finite state accepter) hay DFA được định nghĩa bởi bộ năm:

    $M= (Q,\Sigma, \delta , q_0 ,F)$.

    - ***Q*** là một tập hữu hạn các **trạng thái nội**
    - **Σ** là một tập hữu hạn các ký hiệu được gọi là bảng chữ cái nhập.
    - **δ**: $Q \times \Sigma \rightarrow Q$ là hàm chuyển trạng thái.
    - $\mathbf{q_0} \in Q$ là trạng thái khởi đầu.
    - $\mathbf{F} \in Q$ là một tập trạng thái kết thúc, hay còn gọi là trạng thái chấp nhận.

Chú ý: Automat hữu hạn không có bộ nhớ so với mô hình tổng quát

6.3 Automat hữu hạn đơn định (DFA)

- Hoạt động của một DFA:
  - Tại thời điểm khởi đầu, nó được giả thiết ở trong **trạng thái khởi đầu q_0** với cơ cấu nhập đang ở ký hiệu đầu tiên bên trái của **chuỗi nhập**.
  - Mỗi lần di chuyển, cơ cấu nhập tiến về phía phải 1 ký hiệu và lấy ra.
  - Khi gặp ký hiệu kết thúc, chuỗi được chấp nhận nếu automat đang ở vào 1 trong **các trạng thái chấp nhận được** của nó. Ngược lại thì chuỗi nhập bị từ chối.

6.3 Automat hữu hạn đơn định (DFA)

- Để biểu diễn trực quan cho DFA, dùng đồ thị chuyển trạng thái:
  - Các **đỉnh** biểu diễn các **trạng thái**
  - Các **cạnh** biểu diễn các **chuyển trạng thái**
  - Các **nhãn trên các định** là **tên các trạng thái**
  - Các **nhãn trên các cạnh** là giá trị hiện tại của **ký hiệu nhập**
  - **Trạng thái khởi đầu** sẽ được nhận biết rằng một **mũi tên đi vào** không mang nhãn mà không xuất phát từ định nào.
  - Các **trạng thái kết thúc** được vẽ bằng **vòng tròn đôi**

6.3 Automat hữu hạn đơn định (DFA)

- Cho DFA:

  M= (Q,Σ, δ , q_0 ,F).

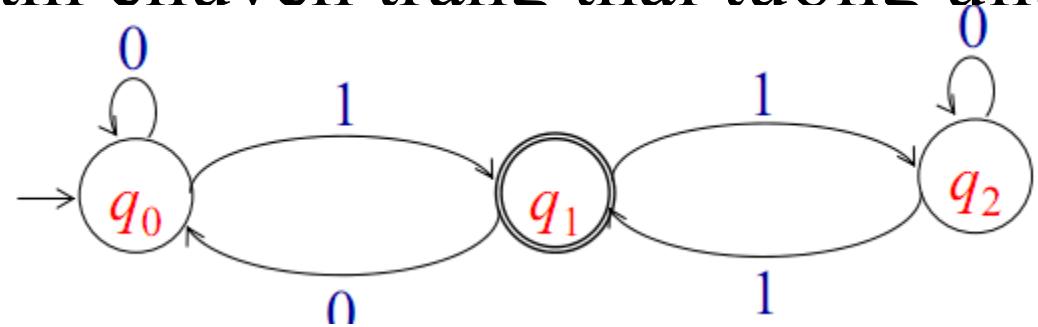
  Q={ q0,q1,q2}, Σ={0,1}, F{q1}, δ được cho bởi:

  $\delta(q_0,0) = q_0$, $\delta(q_0,1) = q_1$

  $\delta(q_1,0) = q_0$ $\delta(q_1,0) = q_2$

  $\delta(q_2,0) = q_2$ $\delta(q_2,0) = q1$

- Đồ thi chuyển trang thái tương ứng



6.3 Automat hữu hạn đơn định (DFA)

- Hàm chuyển trạng thái mở rộng δ^* được định nghĩa đệ quy như sau:
  - $\delta^*(q, \lambda) = q$,
  - $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$, với $q \in Q$, $w \in \Sigma^*$ và $a \in \Sigma$
- Ví dụ:
  - Nếu $\delta(q_0, a) = q_1$ và $\delta(q_1, b) = q_2$
  - Thì $\delta^*(q_0, ab) = q_2$

6.3 Automat hữu hạn đơn định (DFA)

- Định nghĩa 2:
  - Ngôn ngữ được chấp nhận bởi DFA

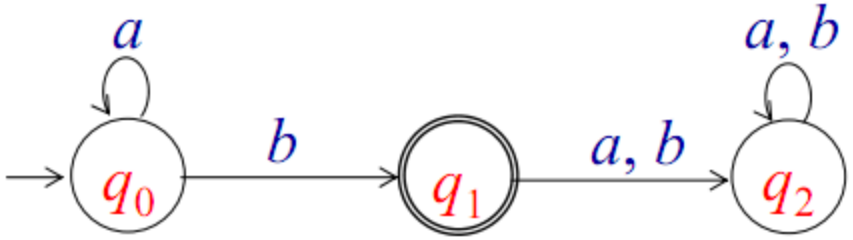
    $M= (Q,\Sigma, \delta , q_0 ,F)$.

    Là tập tất cả các chuỗi trên Σ được chấp nhận bởi M.

  - $L(M)= \{w \in \Sigma^*: \delta^*(q_0 ,w) \in F\}$.

6.3 Automat hữu hạn đơn định (DFA)

- Ví dụ:
  - Xét DFA M sau:



  - DFA trên chấp nhận ngôn ngữ sau:

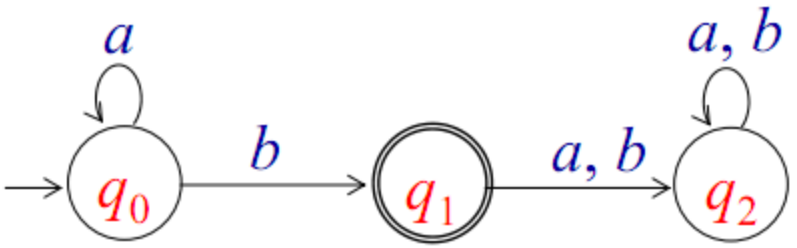
    $L(M)= \{a^n b: n>=0\}$

  - **Trạng thái bẫy**: là trạng thái mà sau khi automat đi vào thì sẽ không bao giờ thoát ra được.

6.3 Automat hữu hạn đơn định (DFA)

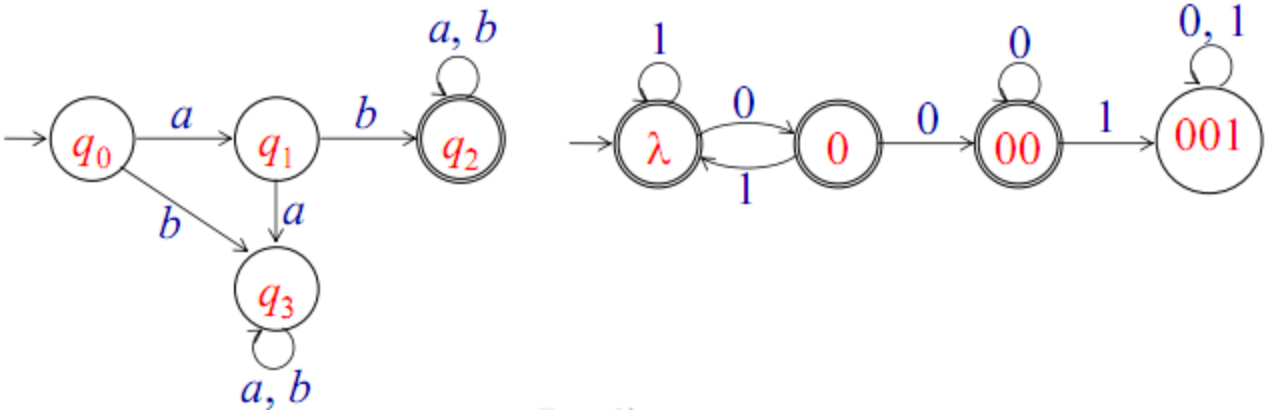
- **Bảng truyền** (transition table)
  - Là bảng trong đó các nhãn của hàng biểu diễn cho trạng thái hiện tại, còn nhãn của cột biểu diễn cho ký hiệu nhập hiện tại. Các điểm nhập trong bảng định nghĩa cho trạng thái kế tiếp.

	a	b
q_0	q_0	q_1
q_1	q_2	q_2
q_2	q_2	q_2



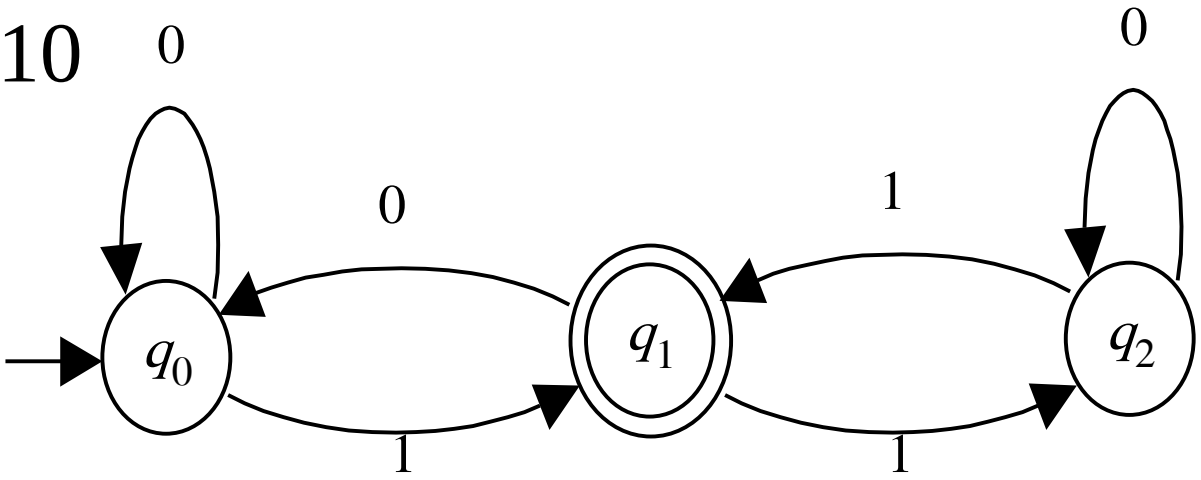
6.3 Automat hữu hạn đơn định (DFA)

- Ví dụ: Tìm DFA chấp nhận ngôn ngữ
  - Tìm DFA M1 chấp nhận tập tất cả các chuỗi trên Σ={a,b} được bắt đầu bằng chuỗi *ab*.
  - Tìm DFA M2 chấp nhận tất cả các chuỗi trên Σ={0,1}, ngoại trừ những chuỗi chứa chuỗi con 001.



6.3 Automat hữu hạn đơn định (DFA)

- Bài tập 1: chuỗi nào được đón nhận: 0001, 01001, 0000110



- Bài tập 2: Xây dựng DFA sao cho
  - Tất cả các chuỗi chỉ có 1 ký tự a
  - Tất cả các chuỗi có ít nhất 1 ký tự a
  - Tất cả các chuỗi không có nhiều hơn 3 ký tự a

6.4 Automat hữu hạn không đơn định (NFA)

- Định nghĩa:

Một Automat hữu hạn không đơn định (nondeterministic finite state accepter) hay **NFA** được định nghĩa bằng bộ năm:

$$M= (Q,\Sigma, \delta , q_0 ,F).$$

Trong đó **Q,Σ, q_0 ,F** được định nghĩa như đối với Accepter hữu hạn đơn định còn **δ** được định nghĩa là:

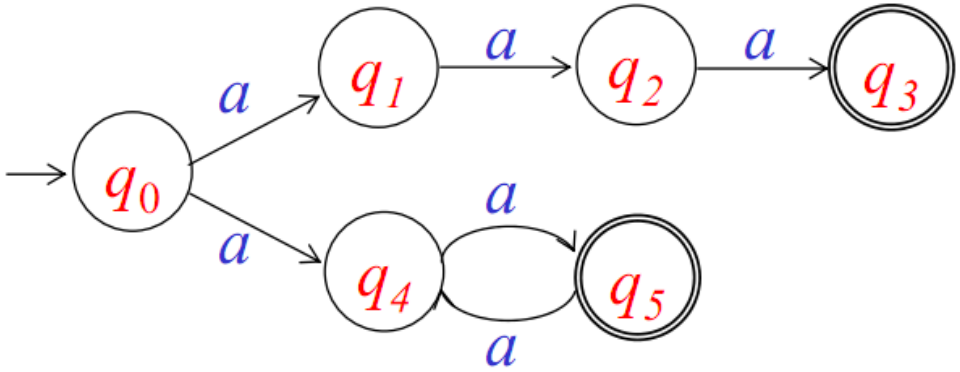
$$\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$$

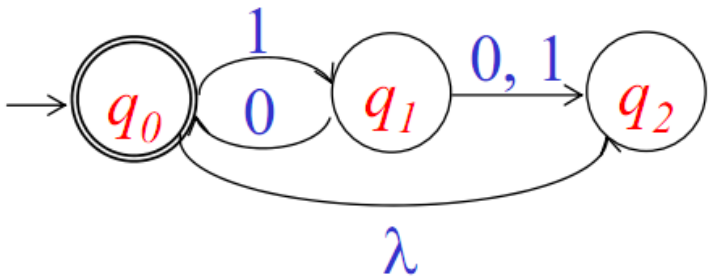
6.4 Automat hữu hạn không đơn định

- Nhận xét:
  - Miền giá trị của δ là tập 2^Q , vì vậy giá trị của nó không còn là 1 phần tử đơn của Q mà là tập con của Q. Ví dụ: $\delta(q_1, a) = \{q_0, q_2\}$. Đặc biệt tập con này có thể là rỗng -> cấu hình chết.
  - λ được coi như đối số thứ 2 của δ nên NFA có thể thực hiện dịch chuyển mà không cần có ký tự nhập.
  - NFA cũng được biểu diễn bằng đồ thị chuyển trạng thái.

6.4 Automat hữu hạn không đơn định

- Ví du:



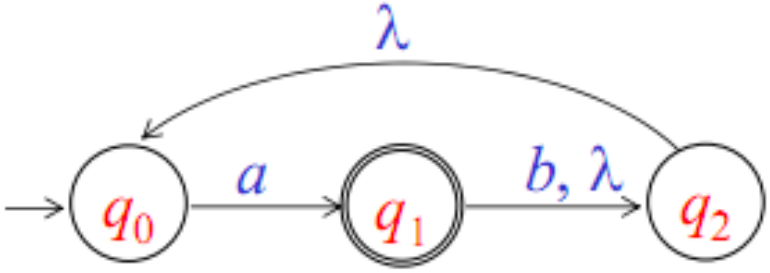


- Hàm chuyển trạng thái mở rộng:
  - Cho một NFA, hàm chuyển trạng thái mở rộng được định nghĩa sao cho $\delta^*(q_i,w)$ chứa q_j nếu và chỉ nếu có một con đường trong đồ thị đi từ q_i đến q_j mang nhãn w. Điều này đúng với mọi q_i, q_j ϵQ và w ϵ Σ^*

6.4 Automat hữu hạn không đơn định

- Ví dụ hàm chuyển mở rộng:

$$\delta^*(q_1, \lambda) = \{q_1, q_2, q_0\}$$
$$\delta^*(q_2, \lambda) = \{q_2, q_0\}$$
$$\delta^*(q_0, a) = \{q_1, q_2, q_0\}$$
$$\delta^*(q_1, a) = \{q_1, q_2, q_0\}$$
$$\delta^*(q_1, b) = \{q_2, q_0\}$$



- Với T là tập con của Q, định nghĩa:

$$\delta(T,a) = \bigcup_{q \in T} \delta(q,a) \qquad \delta^*(T,\lambda) = \bigcup_{q \in T} \delta(q,\lambda) \qquad \delta^*(T,a) = \bigcup_{q \in T} \delta(q,a)$$

6.4 Automat hữu hạn không đơn định

- Bảng truyền NFA:
  - Tập trạng thái S = {0, 1,2, 3}; Σ = {a, b}; Trạng thái bắt đầu s_o = 0; tập trạng thái kết thúc F = {3}.

Trạng thái	Ký hiệu nhập	
	a	b
0	{0, 1}	{0}
1	-	{2}
2	-	{3}

6.4 Automat hữu hạn không đơn định

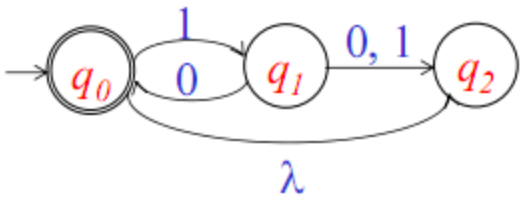
- Ngôn ngữ của NFA:

  Ngôn ngữ được chấp nhận bởi NFA M= (Q,Σ, δ , q_0 ,F), được định nghĩa như một tập các chuỗi được chấp nhận bởi NFA trên. Một cách hình thức:

$$L(M) = \{w \in \Sigma^*: \delta^*(q_0, w) \cap F \neq \varnothing\}.$$

- Ví dụ:
  - Ngôn ngữ được chấp nhận bởi automat dưới là

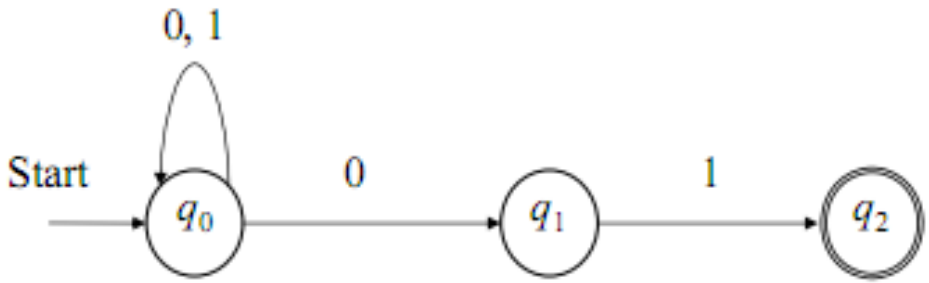
    L={$(10)^n$:n>=0}



6.4 Automat hữu hạn không đơn định

- **Bài tập NFA**: Biểu diễn NFA *N* bằng sơ đồ chuyển và bảng truyền. N chấp nhận ngôn ngữ L={w|w ϵ {0, 1}* và kết thúc bởi 01}

  Σ={q0,q1,q2}, q0, F{q2}.



	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$^*q_2$	$\emptyset$	$\emptyset$

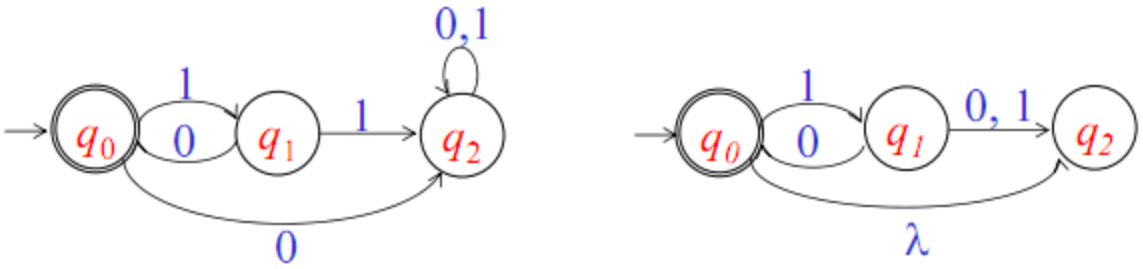
6.5 Chuyển từ NFA sang DFA

6.5.1 Sự tương đương giữa NFA và DFA

6.5.2 Chuyển từ NFA sang DFA

6.5.1 Sự tương đương giữa NFA và DFA

- Sự tương đương giữa hai automat
  - Hai automat được gọi là tương đương nhau nếu chúng cùng chấp nhận một ngôn ngữ như nhau.
- Ví dụ:
  - DFA và NFA sau là tương đương nhau vì cùng chấp nhận ngôn ngữ $\{(10)^n : n>=0\}$



6.5.1 Sự tương đương giữa NFA và DFA

- Nhận xét:
  - DFA bản chất là một loại của NFA.
  - Một NFA thì sẽ có một DFA tương đương với nó.
  - Mọi ngôn ngữ được chấp nhận bởi NFA thì cũng sẽ được chấp nhận bởi DFA.
  - Xây dựng NFA thường dễ dàng hơn.
  - Trong thực tế, số trạng thái của DFA xấp xỉ NFA, nhưng thường có nhiều hàm truyền hơn.
  - Trong trường hợp xấu nhất, nếu cùng chấp nhận một ngôn ngữ: NFA có n trạng thái thì DFA có 2^n trạng thái.

6.5.1 Sự tương đương giữa NFA và DFA

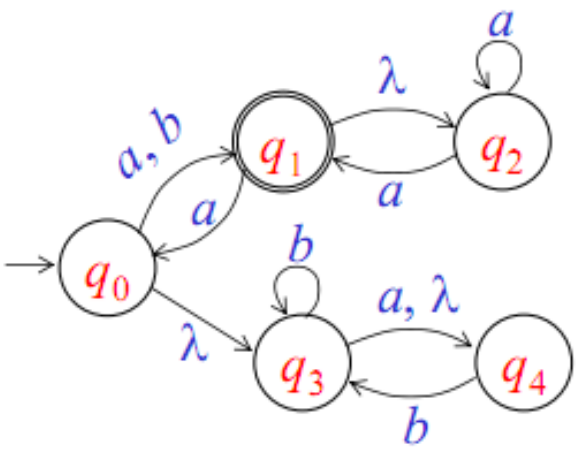
- Định lý về sự tương đương:
  - Cho L là ngôn ngữ được chấp nhận bởi một Automat hữu hạn không đơn định $M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ thì tồn tại một automat hữu hạnh đơn định $M_D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ sao cho $L=L(M_D)$.
- <u>Chú ý:</u>
  - Một trạng thái của NFA là một tập trạng thái của DFA
  - Trạng thái kết thúc của NFA là trạng thái mà có chứa trạng thái kết thúc của DFA.

6.5.2 Chuyển từ NFA sang DFA

- Thủ tục chuyển NFA thành DFA:
  - **Input:** nfa $M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$
  - **Output:** ĐTCTT G_D của DFA M_D
    - B1: Tạo một đồ thị G_D với định khởi đầu là tập $\delta_N^*(q_0\ \lambda)$
    - B2: Lặp lại Bước 3 đến Bước 6 cho đến khi không còn cạnh nào thiếu.
    - B3: Lấy một đỉnh bất kỳ $\{q_i, q_j, \ldots q_k\}$ của G_D mà có 1 cạnh còn chưa được định nghĩa đối với một a nào đó của Σ.
    - B4: Tính $\delta_N^*(\{q_i, q_j, \ldots q_k\}, a) = \{q_l, q_m, \ldots q_n\}$.
    - B5: Tạo một đỉnh cho G_D có nhãn $\{q_l, q_m, \ldots q_n\}$ nếu nó chưa tồn tại.
    - B6: Thêm vào G_D một cạnh từ $\{q_i, q_j, \ldots q_k\}$ đến $\{q_l, q_m, \ldots q_n\}$ và gán nhãn cho nó bằng a.
    - B7: Mỗi trạng thái của G_D mà nhãn của nó chứa một q_f bất kỳ thuộc F_N thì được coi là một đỉnh kết thúc.

6.5.2 Chuyển từ NFA sang DFA

- Ví dụ: Hãy biến đổi NFA dưới thành DFA tương đương



	a	b	λ
q_0	q_1	q_1	q_3
q_1	q_0		q_2
q_2	q_1, q_2		
q_3	q_4	q_3	q_4
q_4		q_3	

6.5.2 Chuyển từ NFA sang DFA

$\delta^*(q_0, \lambda) = \{q_0, q_3, q_4\}$

$\delta^*(\{q_0, q_3, q_4\}, a) = \{q_1, q_2, q_4\}$

$\delta^*(\{q_0, q_3, q_4\}, b) = \{q_1, q_2, q_3, q_4\}$

$\delta^*(\{q_1, q_2, q_4\}, a) = \{q_0, q_1, q_2, q_3, q_4\}$

$\delta^*(\{q_1, q_2, q_4\}, b) = \{q_3, q_4\}$

$\delta^*(\{q_1, q_2, q_3, q_4\}, a) = \{q_0, q_1, q_2, q_3, q_4\}$

$\delta^*(\{q_1, q_2, q_3, q_4\}, b) = \{q_3, q_4\}$

$\delta^*(\{q_0, q_1, q_2, q_3, q_4\}, a) = \{q_0, q_1, q_2, q_3, q_4\}$

$\delta^*(\{q_0, q_1, q_2, q_3, q_4\}, b) = \{q_1, q_2, q_3, q_4\}$

$\delta^*(\{q_3, q_4\}, a) = \{q_4\}$

$\delta^*(\{q_3, q_4\}, b) = \{q_3, q_4\}$

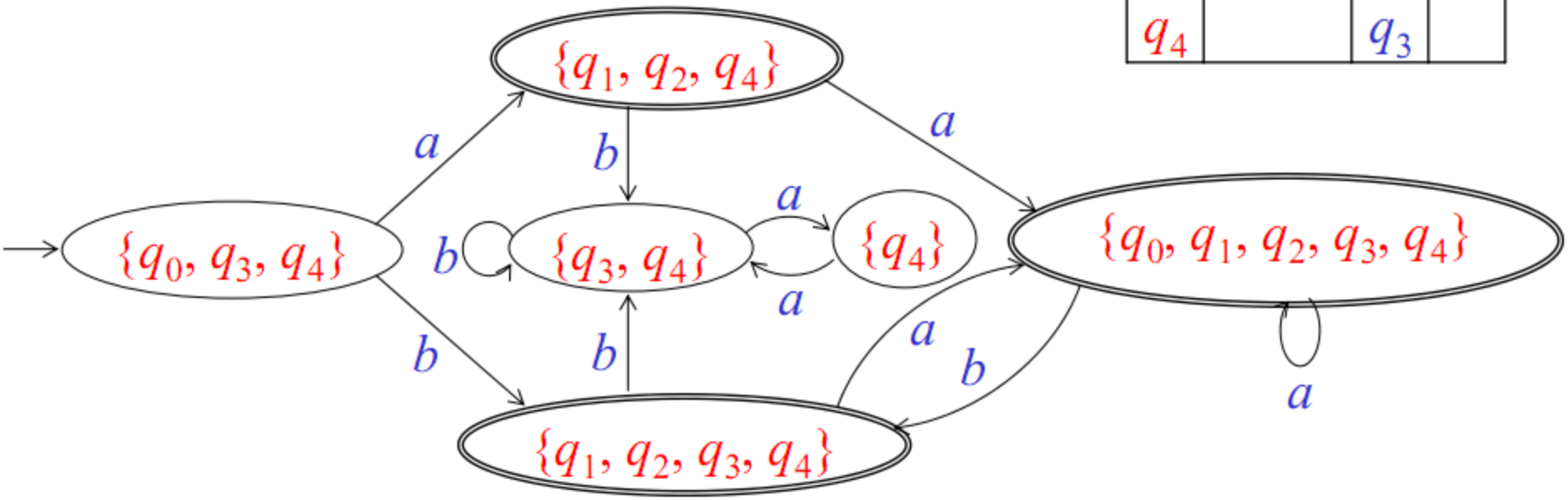
$\delta^*(\{q_4\}, a) = \varnothing$

$\delta^*(\{q_4\}, b) = \{q_3, q_4\}$

	a	b	λ
q_0	q_1	q_1	q_3
q_1	q_0		q_2
q_2	q_1, q_2		
q_3	q_4	q_3	q_4
q_4		q_3	

6.5.2 Chuyển từ NFA sang DFA

	a	b	λ
q_0	q_1	q_1	q_3
q_1	q_0		q_2
q_2	q_1, q_2		
q_3	q_4	q_3	q_4
q_4		q_3	



Kiểm tra

- Biến đổi những NFA sau thành DFA tương đương

Nfa M_1			
	a	b	λ
q_0	q_1	q_3	q_1
q_1	q_2	q_2, q_0	
q_2		q_1	
q_3	q_0, q_4	q_3	q_4
q_4	q_3, q_4	q_4	
$F = \{q_2\}$			

Nfa M_2			
	a	b	λ
q_0	q_1, q_3	q_3	q_3
q_1	q_2	q_2	q_0
q_2	q_1		
q_3	q_4		q_4
q_4	q_4	q_3	
$F = \{q_4\}$			

Nfa M_3			
	a	b	λ
q_0	q_1	q_2	q_1
q_1	q_1, q_2	q_3	q_3
q_2		q_0, q_2	
q_3	q_2, q_3		
$F = \{q_0\}$			

Nội dung

1. Vai trò của bộ phân tích từ vựng
2. Lữu trữ tạm chương trình nguồn
3. Đặc tả Token
4. Nhận dạng Token
5. Sơ đồ dịch
6. Automat hữu hạn
7. **Từ biểu thức chính quy đến NFA**
8. Thiết kế bộ sinh bộ PTTV

7. Từ biểu thức chính quy đến NFA

- Nhắc lại về BTCQ:
  - Mô tả chính xác các từ tố trong NNLT
  - Mỗi loại từ tố được mô tả bằng một BTCQ

a	L(a) = {a} – tập hợp gồm xâu "a"
λ	L(λ) = {λ} – tập hợp gồm xâu rỗng
R|S	L(R|S) = L(R) ∪ L(S) – hợp của L(R) và L(S)
RS	L(RS) = {xy | x ∈ L(R), y ∈ L(S)} – nối 2 xâu bất kì của L(R) và L(S)
R*	L(R*) = L(λ|R|RR|RRR|RRRR …) – nối các xâu của L(R) lại với nhau

7. Từ biểu thức chính quy đến NFA

R+	L(R+) = L(R*) \ {λ} – R* loại bỏ xâu rỗng
R?	L(R?) = L(R|λ)
[abcd]	L([abcd]) = L(a|b|c|d)
[a-z]	L([a-z]) = L(a|b|..|z)
[^abc]	L([^abc]) = kí tự bất kì không thuộc L([abc])
[^a-z]	L([^a-z]) = kí tự bất kì không thuộc L([a-z])

Một số quy ước với BTCQ

7. Từ biểu thức chính quy đến NFA

- Ví dụ BTCQ:

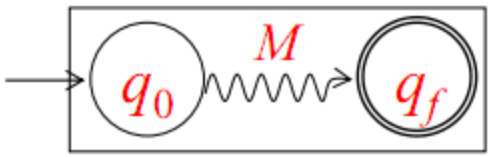
Biểu thức chính quy (RE)	Xâu thuộc ngôn ngữ
a	"a"
ab	"ab"
a | b	"a", "b"
(**ab**)*	"", "ab", "abab" …
(**a** | ε) **b**	"ab", "b"
(a+b.c)*	Λ,a, bc,aa, abc, bca, bcbc,aaa , aabc, …

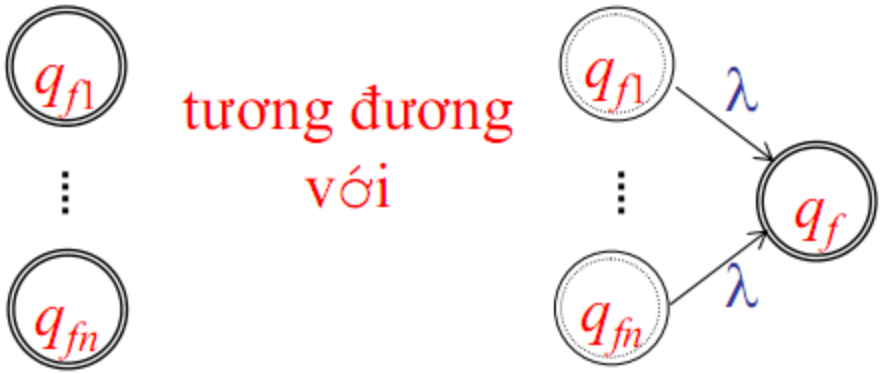
7. Từ biểu thức chính quy đến NFA

- Định nghĩa hình thức cho BTCQ:
  - Cho Σ là một bảng chữ cái, khi đó:
    1. λ và a ϵ Σ tất cả đều là những BTCQ nguyên thủy.
    2. Nếu r1 và r2 là những BTCQ thì r1+r2, r1.r2, $r1^*$ và (r1) cũng là BTCQ
    3. Một chuỗi là một BTCQ nếu và chỉ nếu nó có thể được dẫn xuất từ các BTCQ nguyên thủy bằng một số lần hữu hạn áp dụng 2.

7. Từ biểu thức chính quy đến NFA

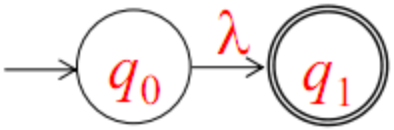
- Định lý:
  - Cho r là một BTCQ, thì tồn tại một NFA chấp nhận L(r).
- Bổ đề:
  - Với mọi NFA có nhiều hơn một trạng thái kết thúc thì luôn luôn có một NFA tương đương với chỉ một trạng thái kết thúc.



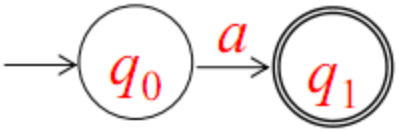


7. Từ biểu thức chính quy đến NFA

- Thủ tục chuyển đổi từ RE sang NFA:
  - **Input**: Biểu thức chính quy r
  - **Output:** NFA M= (Q,Σ, δ , q_0 ,F).
  1. Xây dựng các NFA cho các BTCQ nguyên thủy



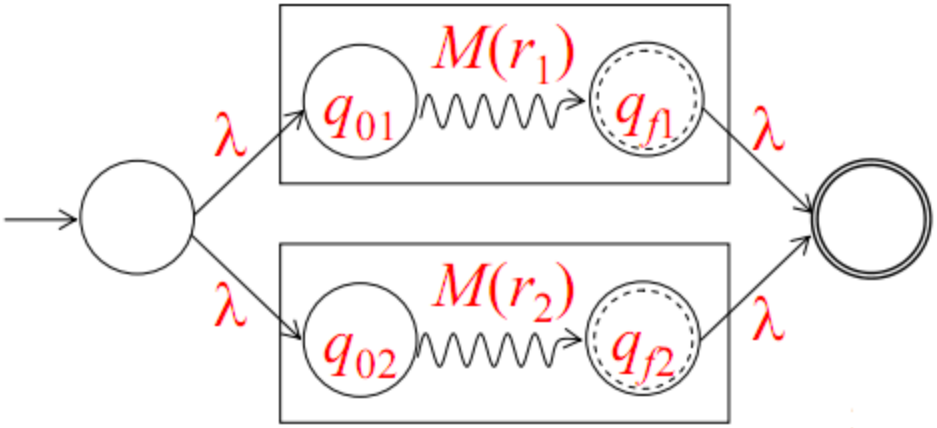
(a) NFA chấp nhận {λ}



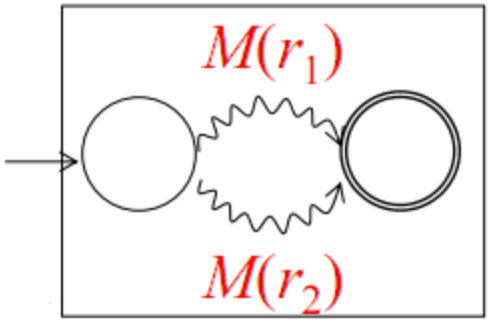
(b) NFA chấp nhận {a}

7. Từ biểu thức chính quy đến NFA

- Thủ tục chuyển đổi từ RE sang NFA:
  2. Xây dựng các NFA cho các BTCQ phức tạp:
     - **NFA cho BTCQ $r_1 + r_2$ $(r_1|r_2)$:** *Giả sử $N(r_1)$ và $N(r_2)$ là NFA cho biểu thức chính quy r_1 và r_2*

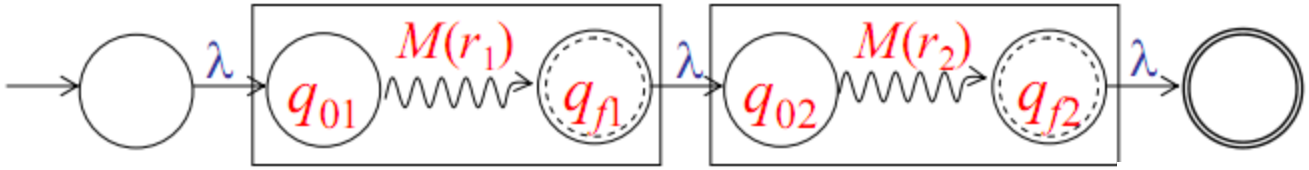


hoặc

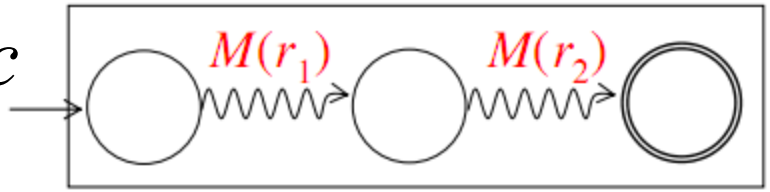


7. Từ biểu thức chính quy đến NFA

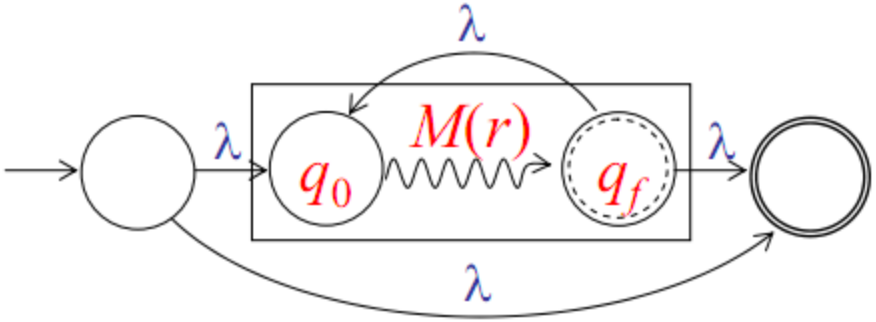
- Xây dựng NFA cho các biểu thức phức tạp:
  - **NFA cho BTCQ $r_1 r_2$:**



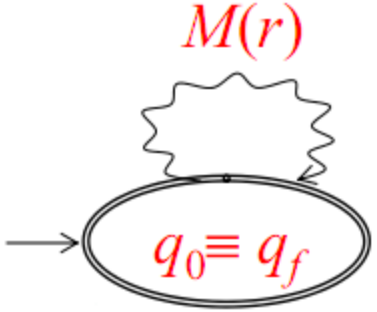
hoặc



  - **NFA cho BTCQ r^*:**



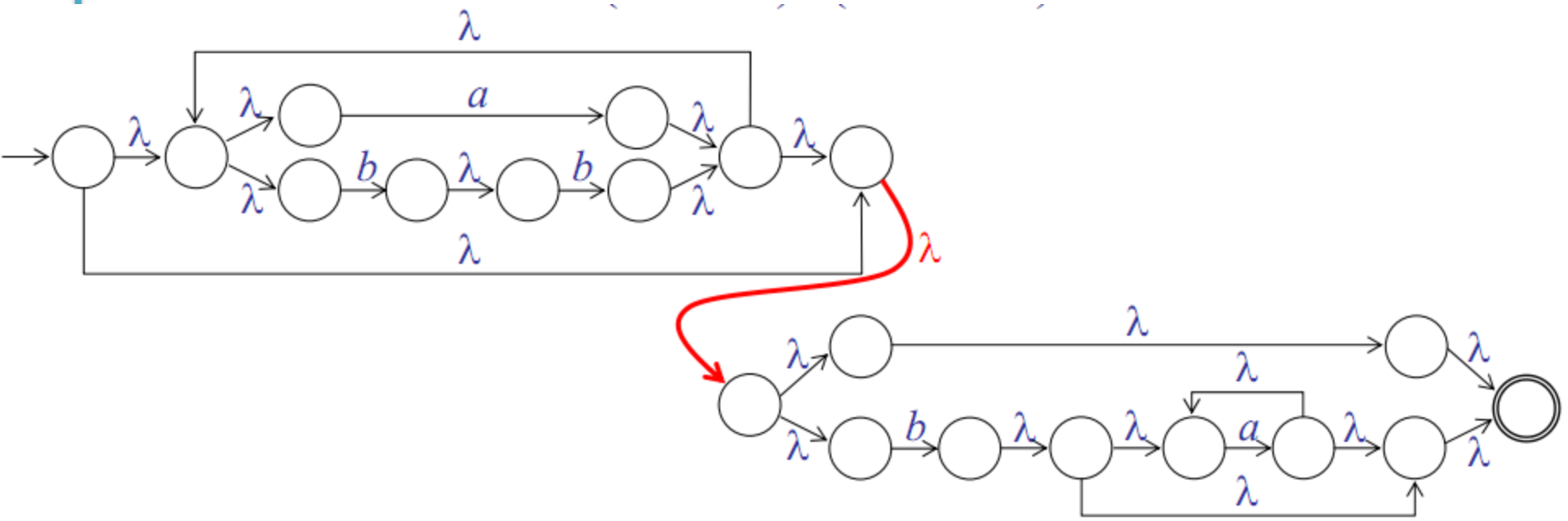
hoặc



7. Từ biểu thức chính quy đến NFA

- Ví dụ: Tìm NFA chấp nhận L(r), trong đó:

$$r = (a + bb)^* (ba^* + \lambda).$$



automat chấp nhận L((a + bb) (ba* + λ).*

7. Từ biểu thức chính quy đến NFA

- Bài tập:
  - Xây dựng NFA cho các BTCQ sau:

$$r_1 = aa^* + aba^*b^*$$
$$r_2 = ab(a + ab)^* (b + aa)$$
$$r_3 = ab^*aa + bba^*ab$$
$$r_4 = a^*b(ab + b)^*a^*$$
$$r_5 = (ab^* + a^*b)(a + b^*a)^* b$$
$$r_6 = (b + a^*)(ba^* + ab)^*(b^*a + ab)$$

Nội dung

1. Vai trò của bộ phân tích từ vựng
2. Lữu trữ tạm chương trình nguồn
3. Đặc tả Token
4. Nhận dạng Token
5. Sơ đồ dịch
6. Automat hữu hạn
7. Từ biểu thức chính quy đến NFA
8. **Thiết kế bộ sinh bộ PTTV**

8. Thiết kế bộ sinh bộ PTTV

- Đặc điểm bộ PTTV:
  - Chương trình PTTV chuyển đổi mã nguồn thành một dãy các từ tố
  - RE có thể mô tả từ tố một cách chính xác
  - RE và thứ tự ưu tiên có thể chuyển thành bộ PTTV qua 2 bước:
    1. Chuyển RE → NFA
    2. Chuyển NFA → DFA (nếu có thể, tối ưu hóa DFA)
  - Kết quả: Bộ PTTV ngắn gọn và dễ bảo trì
- Các chương trình sinh bộ PTTV đã có sẵn và miễn phí. Ví dụ như Lex.

8. Thiết kế bộ sinh bộ PTTV

- Đặc tả chương trình PTTV:
  - Là ***đầu vào*** cho các *chương trình* ***sinh ra*** chương trình phân tích từ vựng
    - Danh sách REs theo thứ tự ưu tiên
    - Hành động gắn liền với mỗi RE khi chương trình PTTV nhận dạng được một từ tố bằng RE đó
  - ***Đầu ra*** của các chương trình này là một chương trình PTTV có thể
    - Đọc chương trình nguồn và tách nó ra thành các từ tố bằng cách nhận dạng REs
    - Thông báo lỗi nếu gặp phải kí tự không đúng theo REs

8. Thiết kế bộ sinh bộ PTTV

- Đặc tả của LEX:

Khai báo
%%

Bao gồm khai báo biến, hằng và các định nghĩa chính quy.

Quy tắc dịch
%%

Có dạng pi {action i }. pi là các biểu thức chính quy, action i là đoạn chương trình mô tả hành động của bộ phân tích từ vựng thực hiện khi pi tương ứng phù hợp với trị từ vựng.

Các thủ tục phụ là sự cài đặt các hành động trong phần 2.

8. Thiết kế bộ sinh bộ PTTV

- **Ví dụ đặc tả chương trình Lex**

```
%%
digits = 0|[1-9][0-9]*
letter = [A-Za-z]
identifier = {letter}({letter}|[0-9_])*
whitespace = [\ \t\n\r]+
%%
{whitespace} {/* discard */}
{digits}     { return new IntegerConstant(Integer.parseInt(yytext()); }
"if"         { return new IfToken(); }
"while"      { return new WhileToken(); }
…
{identifier} { return new IdentifierToken(yytext()); }
```

Tổng kết Bài 3

- Các kiến thức cần nhớ:
  - Các khái niệm cơ bản: Token, trị từ vựng,…
  - Đặc tả và nhận dạng Token
  - Các khái niệm BTCQ, Sơ đồ dịch và Automat
  - Kỹ thuật chuyển đổi trong Automat và BTCQ
  - Quy trình PTTV
  - Khái niệm bộ sinh bộ PTTV.
- Về nhà đọc thêm:
  - Cách tối ưu hóa DFA (sách của Aho)
  - Sử dụng Lex trong Pascal hoặc Java
  - Cài đặt mã nguồn cho các sơ đồ dịch cơ bản bằng C

Bài học phần sau

Bài 4: Phân tích cú pháp

Thảo luận