Cheatsheet: Automata & Lý thuyết Ngôn ngữ Hình thức

Mô Hình Hóa Toán Học - HK241

1. Alphabet và Chuỗi

Alphabet (Σ): Là một tập hợp hữu hạn các ký tự. Ví dụ: $\Sigma = \{0,1\}$.

Chuỗi: Là một dãy ký tự được chọn từ bảng chữ cái Σ. Ví dụ: nếu $\Sigma = \{a,b\}$, thì chuỗi "abba" là một chuỗi hợp lệ.

- **Chuỗi rỗng** (ε): Chuỗi có độ dài bằng 0, không chứa ký tự nào. Độ dài của ε là $|\varepsilon| = 0$.
- **Độ dài của chuỗi** u: Ký hiệu là $|u|$, là số lượng ký tự trong chuỗi. Ví dụ: với $u = abc$, $|u| = 3$.

---

2. Định nghĩa Ngôn ngữ

Ngôn ngữ: Là một tập hợp con của tất cả các chuỗi có thể được tạo ra từ bảng chữ cái Σ, ký hiệu là Σ^* (tập tất cả các chuỗi, bao gồm cả chuỗi rỗng).

- **Ngôn ngữ rỗng** ($\emptyset$): Là tập hợp không chứa bất kỳ chuỗi nào.
- **Ngôn ngữ vũ trụ** (Σ^*): Là tập hợp chứa tất cả các chuỗi có thể có trên bảng chữ cái Σ, bao gồm cả chuỗi rỗng.

---

3. Phép Toán Trên Chuỗi

Phép kết chuỗi (Concatenation): Phép kết hợp hai chuỗi u và v, ký hiệu là $u.v$, tạo ra một chuỗi mới bằng cách nối v vào phía sau u.

- Ví dụ: Nếu $u = 01$ và $v = 10$, thì $u.v = 0110$. Nếu $u = \varepsilon$ và $v = 110$, thì $u.v = 110$.

---

4. Đặc tả Ngôn ngữ (Specifying Languages)

Một ngôn ngữ có thể được đặc tả bằng nhiều cách:

- **Liệt kê chuỗi**: Liệt kê trực tiếp các chuỗi. Ví dụ:
  - $L_1 = \{\varepsilon, 0, 1\}$
  - $L_2 = \{a, ab, abc\}$
  - $L_3 = \{w \in \{a,b\}^* : |w|_a = |w|_b\}$
- **Đặc tính**: Mô tả tính chất mà tất cả các chuỗi trong ngôn ngữ đều có. Ví dụ:
  - $L_4 = \{w \in \{0,1\}^* : n_0(w) = n_1(w)\}$ với $n_a(w)$ là số lần xuất hiện của a trong w.
  - $L_5 = \{w \in \{a,b\}^* : |w|$ chẵn$\}$.
- **Biểu thức chính quy** (Regular expression): Mô tả một tập hợp các chuỗi.
  - a^*: Tất cả các chuỗi gồm nhiều ký tự 'a', bao gồm cả chuỗi rỗng.
  - $(a|b)^*$: Tất cả các chuỗi có thể được tạo từ các ký tự 'a' và 'b'.
- **Văn phạm (Grammar)**: Sử dụng quy tắc sinh để tạo ra các chuỗi trong ngôn ngữ. Văn phạm bao gồm:
  - **Ký hiệu không kết thúc** (Non-terminal symbols, ký hiệu là N): Đây là các ký hiệu trung gian, chưa phải chuỗi hoàn chỉnh, được sử dụng trong quá trình sinh chuỗi. Ví dụ: S.
  - **Ký hiệu kết thúc** (Terminal symbols, ký hiệu là T): Là các ký hiệu cuối cùng trong chuỗi đầu ra. Đây là những ký hiệu không thể tiếp tục thay thế được. Ví dụ: a, b, c.
  - **Ký hiệu bắt đầu** (S): Là một ký hiệu không kết thúc được chọn làm điểm khởi đầu để sinh chuỗi.
  - **Quy tắc sinh** (Production rules): Các quy tắc xác định cách các ký hiệu không kết thúc có thể được thay thế bằng một chuỗi khác gồm các ký hiệu kết thúc và không kết thúc. Ví dụ:
    - $S \to aSb \,|\, \varepsilon$ (Tạo ra các chuỗi cân bằng giữa 'a' và 'b').

Ví dụ chi tiết về quy tắc sinh

1. Văn phạm cho chuỗi đối xứng

- **Ký hiệu không kết thúc**: S
- **Ký hiệu kết thúc**: a, b
- **Quy tắc sinh**:

$$S \to aSa \,|\, bSb \,|\, \varepsilon$$

Giải thích:

- Quy tắc $S \to aSa$ có nghĩa là nếu bạn có một ký hiệu S, bạn có thể thay thế nó bằng ký tự a, tiếp theo là một ký hiệu S khác, và cuối cùng là ký tự a nữa, tạo ra chuỗi đối xứng.
- Tương tự, quy tắc $S \to bSb$ cho phép bạn tạo ra chuỗi đối xứng bằng ký tự b.
- Quy tắc $S \to \varepsilon$ cho phép chuỗi rỗng là một phần của ngôn ngữ.

Chuỗi được tạo ra: Từ quy tắc trên, ta có thể tạo ra các chuỗi như:

- (từ $S \to aSa \to a\varepsilon a$)
- $aabbaa$ (từ $S \to aSa \to aaSaa \to aabSbaa$)
- $abba$ (từ $S \to aSa \to abSb \to ab\varepsilon b$)

2. Văn phạm cho chuỗi có số lượng ký tự 'a' và 'b' bằng nhau

- **Ký hiệu không kết thúc**: S
- **Ký hiệu kết thúc**: a, b
- **Quy tắc sinh**:

$$S \to aSb \,|\, bSa \,|\, SS \,|\, \varepsilon$$

Giải thích:

- Quy tắc $S \to aSb$ cho phép ta thêm một ký tự a ở đầu và một ký tự b ở cuối, duy trì số lượng ký tự a và b bằng nhau.
- Tương tự, quy tắc $S \to bSa$ cho phép ta thêm ký tự b ở đầu và a ở cuối.
- Quy tắc $S \to SS$ cho phép ta kết hợp các chuỗi con từ S.
- Quy tắc $S \to \varepsilon$ cho phép chuỗi rỗng.

Chuỗi được tạo ra: Từ quy tắc trên, ta có thể tạo ra các chuỗi như:

- ab (từ $S \to aSb \to a\varepsilon b$)
- $aabb$ (từ $S \to SS \to aSbS \to a\varepsilon b\varepsilon$)
- $abab$ (từ $S \to aSb \to abS \to ab\varepsilon$)

---

5. Phép Toán Trên Ngôn ngữ (Operations on Languages)
Cho L_1, L_2 là các ngôn ngữ trên bảng chữ cái Σ:
Phép hợp (Union): $L_1 \cup L_2 = \{w | w \in L_1 \text{ hoặc } w \in L_2\}$
Phép giao (Intersection): $L_1 \cap L_2 = \{w | w \in L_1 \text{ và } w \in L_2\}$
Phép hiệu (Difference): $L_1 - L_2 = \{w | w \in L_1 \text{ và } w \notin L_2\}$
Phép bù (Complement): $\overline{L_1} = \{w | w \in \Sigma^* \text{ và } w \notin L_1\}$
Phép kết nối (Concatenation): $L_1L_2 = \{uv | u \in L_1 \text{ và } v \in L_2\}$
Phép lũy thừa (Power):

- $L^0 = \{\varepsilon\}$
- $L^{n+1} = LL^n$, với $n \geq 0$

Phép sao Kleene (Kleene star): $L^* = \bigcup_{n \geq 0} L^n$ (bao gồm cả ε)
Phép cộng Kleene (Kleene plus): $L^+ = \bigcup_{n \geq 1} L^n$ (không bao gồm ε)
Các phép toán hợp, giao và kết nối là các phép toán chính quy (regular operations).

---

6. Định lý Kleene: Một ngôn ngữ L được gọi là chính quy nếu và chỉ nếu tồn tại một biểu thức chính quy biểu diễn ngôn ngữ L.
Trường hợp 1: Ngôn ngữ L_1 gồm các chuỗi nhị phân có số 0 chẵn. Biểu thức chính quy biểu diễn ngôn ngữ này là:

$$L_1 = (00)^*$$

Trường hợp 2: Ngôn ngữ L_2 gồm các chuỗi nhị phân bắt đầu bằng 1 và kết thúc bằng 0. Biểu thức chính quy biểu diễn ngôn ngữ này là:

$$L_2 = 1(0+1)^*0$$

Trường hợp 3: Xét ngôn ngữ L được định nghĩa như sau:

$$L = a^n b^n c^n \mid n \geq 0$$

Ngôn ngữ này bao gồm các chuỗi có số lượng ký tự 'a', 'b', và 'c' bằng nhau, theo thứ tự đó. **Các chuỗi thuộc L:**

- ε (chuỗi rỗng)
- abc
- $aabbcc$
- $aaabbbccc$
- ...

Chứng minh L không phải là ngôn ngữ chính quy: Chúng ta sẽ sử dụng định lý Pumping để chứng minh rằng L không phải là ngôn ngữ chính quy.

Giả sử L là ngôn ngữ chính quy. Theo định lý Pumping, tồn tại một số $p > 0$ sao cho mọi chuỗi $s \in L$ có $|s| \geq p$ có thể được chia thành $s = xyz$, thỏa mãn:

- $|xy| \leq p$
- $|y| > 0$
- $xy^iz \in L$ với mọi $i \geq 0$

Chọn $s = a^p b^p c^p \in L$. Rõ ràng $|s| = 3p \geq p$. Do $|xy| \leq p$, y chỉ có thể chứa các ký tự 'a', hoặc các ký tự 'a' và 'b'. Xét trường hợp $i = 2$:

- Nếu y chỉ chứa 'a', thì xy^2z sẽ có dạng $a^{p+k}b^pc^p$, với $k > 0$.
- Nếu y chứa cả 'a' và 'b', thì xy^2z sẽ có dạng $a^{p+k_1}b^{p+k_2}c^p$, với $k_1, k_2 > 0$.

Trong cả hai trường hợp, $xy^2z \notin L$ vì số lượng 'a', 'b', và 'c' không bằng nhau. Điều này mâu thuẫn với giả thiết ban đầu rằng L là ngôn ngữ chính quy.

Kết luận: Do mâu thuẫn này, chúng ta kết luận rằng giả thiết ban đầu là sai. Vì vậy, L không phải là ngôn ngữ chính quy. **Giải thích:** Ngôn ngữ này đòi hỏi khả năng "đếm" và so sánh số lượng của ba ký tự khác nhau, điều mà một automat hữu hạn không thể thực hiện được. Đây là một đặc điểm chung của nhiều ngôn ngữ không chính quy - chúng thường yêu cầu một dạng "bộ nhớ" vô hạn để theo dõi và so sánh số lượng các ký tự khác nhau.

---

7. Automata hữu hạn (FA)

- Gồm các thành phần:
  - Tập các trạng thái hữu hạn Q
  - Một trạng thái khởi đầu $q_0 \in Q$
  - Tập các trạng thái kết thúc $F \subseteq Q$
  - Hàm chuyển trạng thái δ
- Số trạng thái là hữu hạn

Loại Automata hữu hạn

- **NFA (Nondeterministic Finite Automaton)**:
  - Một trạng thái có thể chuyển đến ≥ 1 trạng thái khác
  - Hàm chuyển: $\delta : Q \times \Sigma \to \mathscr{P}(Q)$
- **DFA (Deterministic Finite Automaton)**:
  - Mỗi trạng thái chuyển đến đúng một trạng thái khác
  - Hàm chuyển: $\delta : Q \times \Sigma \to Q$
- **ε-NFA**:
  - NFA cho phép chuyển trạng thái mà không đọc kí tự đầu vào
  - Hàm chuyển: $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathscr{P}(Q)$

---

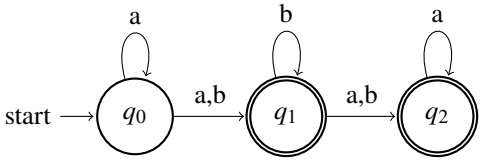
8. Chuyển đổi giữa các dạng Automata

Automata hữu hạn sang Biểu thức chính quy

Phương pháp loại bỏ trạng thái (State Elimination):

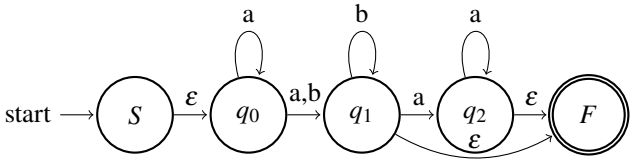
- Quy tắc 1: Không có cạnh vào ở trạng thái khởi đầu
- Quy tắc 2: Không có cạnh ra ở trạng thái kết thúc
- Quy tắc 3: Chỉ có 1 trạng thái kết thúc duy nhất
- Quy tắc 4: Loại bỏ trạng thái trung gian một cách lần lượt

Cho một automata như sau:

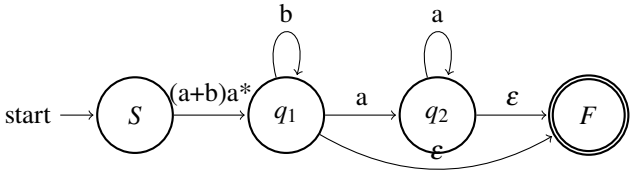


- automata chưa thỏa mãn quy tắc 1. Có cạnh vào a ở trạng thái bắt đầu q_0
- automata chưa thỏa mãn quy tắc 2. Có cạnh ra a ở trạng thái chấp nhận q_2
- automata chưa thỏa mãn quy tắc 3. Có 2 trạng thái chấp nhận

Biến đổi 1: Thỏa mãn 3 quy tắc trên như sau:



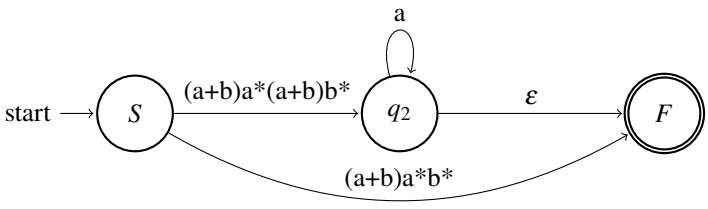
Bước 2:



Eliminate q_0

in	out
S	q_1

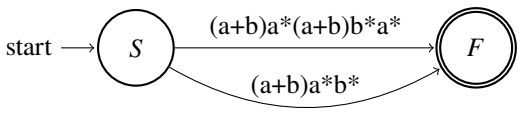
Bước 3:



Eliminate q_1

in	out
S	q_2, F

Bước 4: Tương tự với loại bỏ q2



Vậy mã RE là: (a+b)a*(a+b)b*a* + (a+b)a*b*

9. Chuỗi chuyển đổi

ε-NFA $\rightarrow$ NFA $\rightarrow$ DFA $\rightarrow$ minDFA

Chuyển đổi ε-NFA sang NFA Thuật toán

1. Tính ε-closure cho mỗi trạng thái:
   - ε-closure$(q) = \{p \mid q \xrightarrow{\varepsilon^*} p\}$
2. Xây dựng hàm chuyển trạng thái mới δ':
   - $\delta'(q,a) = \bigcup_{p \in \varepsilon\text{-closure}(q)} \varepsilon\text{-closure}(\delta(p,a))$
3. Xác định trạng thái kết thúc mới:
   - $F' = \{q \mid \varepsilon\text{-closure}(q) \cap F \neq \emptyset\}$

Lưu ý:

$|Q_{\varepsilon\text{-NFA}}| = n \Rightarrow |Q_{NFA}| = n$

CÓ NHIỀU THUẬT TOÁN CHUYỂN ĐỔI, NẾU CÁC BẠN DÙNG THUẬT TOÁN ε-closure CÓ THỂ LÊN YOUTUBE TÌM VIDEO CỦA THẦY MAI XUÂN TOÀN Ở PHẦN ÔN TẬP GIỮA KỲ ĐỂ NGHE GIẢNG

https://youtu.be/5tVrYj8xtsk?list=PLTpNwHSD94uuZ42lCvOiuSleCTivxGPhF

Chuyển đổi NFA sang DFA

Thuật toán đơn định hóa (Determinization Algorithm)

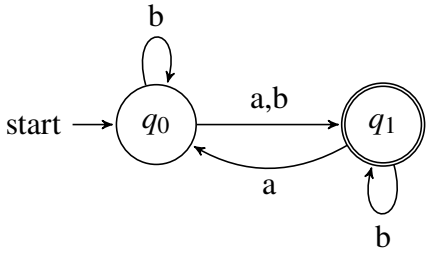
1. Khởi tạo:
   - $Q_D = \{\{q_0\}\}$ (trạng thái khởi đầu của DFA)
   - $q_{0D} = \{q_0\}$
2. Xây dựng hàm chuyển trạng thái δ_D:
   - Với mỗi $S \in Q_D$ và $a \in \Sigma$:
     - $\delta_D(S,a) = \bigcup_{q \in S} \delta(q,a)$
     - Nếu $\delta_D(S,a) \notin Q_D$, thêm vào Q_D
3. Xác định trạng thái kết thúc:
   - $F_D = \{S \in Q_D \mid S \cap F \neq \emptyset\}$
4. Đơn định hóa trạng thái:
   - Loại bỏ các trạng thái trùng lặp và không cần thiết trong Q_D bằng cách áp dụng một phương pháp đơn định hóa.
   - Kết quả cuối cùng là một DFA hoàn chỉnh với tất cả các trạng thái cần thiết và không có trạng thái thừa.

Lưu ý:

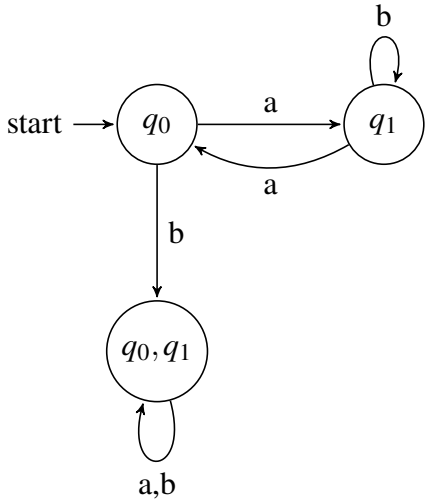
- $Q_{NFA} = \{q_0, q_1, q_2, \ldots, q_n\}$
- $Q_{DFA} \subseteq P(Q_{NFA})$
  $|Q_{DFA}| = n \Rightarrow |Q_{DFA}| \leq 2^n$

Ví dụ

Xét NFA sau:



δ	a	b
$\rightarrow q_0$	q_1	q_0, q_1
$*q_1$	q_0	q_1



δ'	a	b
$\rightarrow q_0$	q_1	q_0q_1
q_1	q_0	q_1
q_0q_1	q_0q_1	q_0q_1

Quy trình chuyển đổi từ ε-NFA sang DFA:

1. Tính ε-closure cho tất cả các trạng thái:

$$\varepsilon\text{-closure}(q) = p \mid q \xrightarrow{\varepsilon^*} p$$

2. Xác định trạng thái khởi đầu của DFA:

$$q_0^{\text{DFA}} = \varepsilon\text{-closure}(q_0^{\varepsilon\text{-NFA}})$$

3. Xây dựng bảng chuyển trạng thái của DFA:

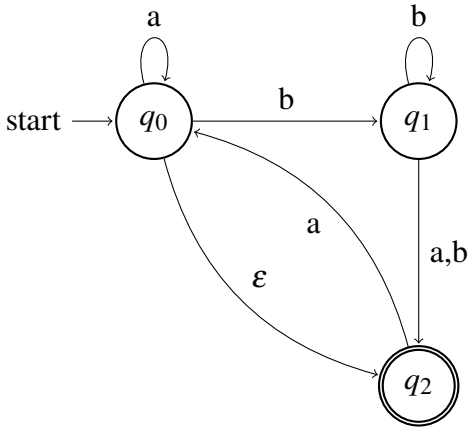
$$\delta'(A,a) = \varepsilon\text{-closure}(\delta(A,a))$$

trong đó $\delta(A,a) = \bigcup_{q\in A} \delta(q,a)$

4. Xác định các trạng thái kết thúc của DFA:

$$F^{\text{DFA}} = \{A \mid A \cap F^{\varepsilon\text{-NFA}} \neq \emptyset\}$$

Ví dụ: Xét ε-NFA sau:



δ	a	b
$\rightarrow q_0$	q_0	q_1
q_1	q_2	q_1, q_2
q_2	q_0	-

q_i	ε-closure
q_0	q_0, q_1
q_1	q_1
q_2	q_2

Áp dụng:

$$\delta'(A,a) = \varepsilon\text{-closure}(\delta(A,a))$$

δ'	a	b
$\rightarrow \{q_0, q_2\}$	$\{q_0q_2\}$	$\{q_1\}$
$\{q_1\}$	$\{q_2\}$	$\{q_1q_2\}$
$\{q_2\}$	$\{q_0q_2\}$	-
$\{q_1, q_2\}$	$\{q_0q_2\}$	$\{q_1q_2\}$

Xác định giá trị kết thúc: Step kết thúc của DFA là những step chứa step kết thúc của εNFA. Ở ví dụ trên là q_0q_2, q_2 vàq_1q_2
Từ bảng trên ta có thể vẻ được 1 DFA tương đương với $\varepsilon - NFA$ ban đầu.

Tối thiểu hóa DFA
Thuật toán Myhill-Nerode

1. Loại bỏ trạng thái không đạt được:
   - Tìm tất cả trạng thái có thể đạt được từ trạng thái khởi đầu
   - Loại bỏ các trạng thái không đạt được
2. Phân hoạch ban đầu:
   - $\pi_0 = \{F, Q \setminus F\}$ (trạng thái final và non-final)
3. Tinh chỉnh phân hoạch:
   - Với mỗi cặp trạng thái (p,q) trong cùng một lớp:
   - Nếu $\exists a \in \Sigma : \delta(p,a)$ và $\delta(q,a)$ thuộc các lớp khác nhau
   - Tách p và q thành các lớp riêng biệt
4. Lặp lại bước 3 cho đến khi không có sự thay đổi

5. Xây dựng DFA tối thiểu: Mỗi lớp trong phân hoạch cuối cùng trở thành một trạng thái

Bước 1: Lập bảng phân loại ban đầu

Ban đầu, các trạng thái được phân chia thành nhóm trạng thái final và non-final. Các trạng thái giống nhau về tính chất được đánh số giống nhau.

Bước 2: Tính cl(s,a), cl(s,b) cho các trạng thái

Ở bước này, ta đã tính toán cl(s,a), cl(s,b), cl(s,c) cho từng trạng thái. Dựa trên sự giống nhau của các cột, ta cập nhật lại giá trị 'cl mới'.

Bước 3: Lặp lại quá trình phân chia

Ta tiếp tục lặp lại quá trình tính toán và so sánh các giá trị cl cho đến khi không còn thay đổi.

Bước 4: Kết luận

Số trạng thái khác nhau thu được là x, đây là số trạng thái trong DFA tối thiểu cần tìm. Các trạng thái của DFA tối thiểu sẽ tương ứng với các lớp phân hoạch cuối cùng.