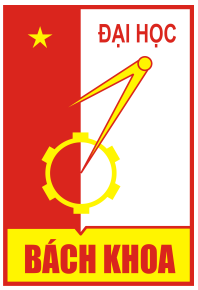
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC



NGÔN NGỮ CHÍNH QUY, BIỂU THỨC CHÍNH QUY VÀ SƠ LƯỢC VỀ LÝ THUYẾT MÃ

Giảng viên hướng dẫn: TS. Ngô Thị Hiền

Nhóm Sinh viên thực hiện:

Nguyễn Anh Tú

Phạm Anh Tuấn

Lớp: KSTN Toán Tin K60

HÀ NỘI - 12/2018

Mục lục

Chương 1

Giới thiệu

Ngôn ngữ là phương tiện để giao tiếp, sự giao tiếp có thể hiểu là giao tiếp giữa con người với nhau, giao tiếp giữa người với máy, hay giao tiếp giữa máy với máy. Ngôn ngữ để con người có thể giao tiếp với nhau được gọi là ngôn ngữ tự nhiên, chẳng hạn như tiếng Anh, tiếng Việt... là các ngôn ngữ tự nhiên. Các quy tắc cú pháp của ngôn ngữ tự nhiên nói chung rất phức tạp nhưng các yêu cầu nghiêm ngặt về ngữ nghĩa thì lại thiếu chặt chẽ, chẳng hạn cùng một từ hay cùng một câu ta có thể hiểu chúng theo những nghĩa khác nhau tùy theo từng ngữ cảnh cụ thể. Con người muốn giao tiếp với máy tính tất nhiên cũng thông qua ngôn ngữ. Để có sự giao tiếp giữa người với máy hay giữa máy với nhau, cần phải có một ngôn ngữ với các quy tắc cú pháp chặt chẽ hơn so với các ngôn ngữ tự nhiên, nói cách khác, với một từ hay một câu thì ngữ nghĩa của chúng phải là duy nhất mà không phụ thuộc vào ngữ cảnh. Những ngôn ngữ như thế được gọi là ngôn ngữ hình thức. Con người muốn máy tính thực hiện công việc, phải

viết các yêu cầu đưa cho máy bằng ngôn ngữ máy hiểu được. Việc viết các yêu cầu như thế gọi là lập trình. Ngôn ngữ dùng để lập trình được gọi là ngôn ngữ lập trình. Các ngôn ngữ lập trình đều là các ngôn ngữ hình thức.

Trong bài báo cáo này, nhóm em sẽ trình bày về ngôn ngữ chính quy, biểu thức chính quy và sơ lược về lý thuyết mã. Ngôn ngữ chính quy (regular languages) thường được biết đến nhờ ứng dụng trong việc tìm từ trong văn bản hay trong trình dịch. Tuy nhiên, trong khuôn khổ của bài báo cáo này, nhóm chúng em sẽ tập trung trình bày những tính chất của ngôn ngữ chính quy và đi giải quyết các bài toán liên quan đến việc biểu diễn và cực tiểu hóa các cách biểu diễn của ngôn ngữ chính quy. Về nội dung của phần lý thuyết mã, bài báo cáo sẽ trình bày những kết quả cơ bản trong lý thuyết mã và một ứng dụng quan được sử dụng rỗng rãi trong trong việc truyền thông tin đó là mã sửa sai.

Chương 2

Kiến thức cơ sở

2.1 Xâu

Bảng chữ cái là một tập hợp các kí tự (có thể gồm hữu hạn hoặc vô hạn các phần tử). Xâu là một dãy hữu hạn các kí tự thuộc bảng chữ cái. Độ dài của xâu là số các kí tự trong xâu đó. Xấu rỗng, được kí hiệu là ϵ, độ dài xâu rồng bằng 0.

Xâu s được gọi là xâu con của w nếu $\exists s_0, s_1$ sao cho $s_0 s s_1 = w$. Đặc biệt:

- Nếu $s_0 = \epsilon$ thì s được gọi là prefix của s
- Nếu $s_1 = \epsilon$ thì s được gọi là suffix của s

Các phép toán trên xâu

1. Tích ghép Cho 2 xâu: $a = a_1 a_2 ... a_n, b = b_1 b_2 ... b_m$ trên bảng chữ cái A. Tích ghép của hai xâu a, b là xâu $c = ab = a_1 1_2 ... a_n b_1 b_2 ... b_m$

   Nhận xét: Cho các xâu s, r, w trên bảng chữ cái A

- Xâu rỗng là phần tử đơn vị với phép nối xâu

  $s\epsilon = \epsilon s = s$

- Phép ghép nối có tính chất kết hợp

  $(sr)\, w = s\,(rw)$

- Kí hiệu w^n, với n là số tự nhiên

$$w^n = \begin{cases} \epsilon, n = 0 \\ w, n = 1 \\ w^{n-1}w, n > 1 \end{cases}$$

2. Phép đảo ngược xâu

   w^R được gọi là xâu đảo ngược của w nếu:

$$w^R = \begin{cases} \epsilon, w = \epsilon \\ s_n s_{n-1} ... s_0, w = s_0 ... s_{n-1} s_n; s_i \in A, i = \overline{1,n} \end{cases}$$

   Cho xâu s, w. Phép đảo ngược xâu có các tính chất sau:

   - $\left(w^R\right)^R = w$
   - $(sw)^R = w^R s^R$
   - $\left|w^R\right| = |w|$

2.2 Ngôn ngữ

Ngôn ngữ là tập các xâu trên một bảng chữ cái.

Các phép toán trên ngôn ngữ

Xét hai ngôn ngữ L_1, L_2 trên bảng chữ cái A

1. Phép tích ghép

   Cho ngôn ngữ L_1 trên bảng chữ cái A_1, L_2 trên bảng chữ cái A_2. Phép nhân ghép của hai ngôn ngữ L_1, L_2 là một ngôn ngữ trên bảng chữ cái $A_1 \cup A_2$, kí hiệu L_1L_2:

   $$L_1L_2 = \{ab | a \in L_1, b \in L_2\}$$

   Nhận xét: Xét các ngôn ngữ L_1, L_2, L_3.Phép nhân ghép có các tính chất sau

   - Tính chất kết hợp: $(L_1L_2)\, L_3 = L_1\, (L_2L_3)$
   - $\forall L_1 : L_1\varnothing = \varnothing L_1 = \varnothing$
   - Tính chất phân phối đối với phép nhân ghép và phép hợp:

     $$(L_1 \cup L_2)\, L_3 = (L_1L_3) \cup (L_2L_3)$$
     $$L_1\, (L_2 \cup L_3) = (L_1L_3) \cup (L_2L_3)$$

2. Phép toán sao

   Ta định nghĩa L^n là phép tích ghép của n lần ngôn ngữ L

   $$L^n = \prod_{i=0}^{n} L$$

   Trong đó $L^0 = \epsilon$.Khi đó, phép toán sao trên ngôn ngữ L được định nghĩa như sau

   $$L^* = \bigcup_{i=0}^{\infty} L^i$$

Ta cũng định nghĩa

$$L^+ = L^* \setminus \{\epsilon\}$$

Ở đây, ta có chú ý rằng $A*$ là một vị nhóm tự do đối với phép toán tích ghép.

3. Phép hợp

   Hợp của hai ngôn ngữ L_1, L_2, kí hiệu $L_1 \cup L_2$ là một ngôn ngữ trên bảng chữ cái A:

$$L_1 \cup L_2 = \{w \in A^* | w \in L_1 \text{ hoặc } w \in L_2\}$$

   Nhận xét: Xét các ngôn ngữ L_1, L_2, L_3.Phép hợp có các tính chất sau

   - Tính chất giao hoán: $L_1 \cup L_2 = L_2 \cup L_1$
   - Tính chất kết hợp: $(L_1 \cup L_2) \cup L_3 = L_1 \cup (L_2 \cup L_3)$
   - $\forall L1 : L_1 \cup \varnothing = \varnothing \cup L1 = L1$ và $L_1 \cup A^* = A^*$

4. Phép giao

   Giao của hai ngôn ngữ L_1, L_2, kí hiệu $L_1 \cap L_2$

$$L_1 \cap L_2 = \{w \in A^* | w \in L_1 \text{ và } w \in L_2\}$$

   Nhận xét: Xét các ngôn ngữ L_1, L_2, L_3.Phép hợp có các tính chất sau

   - Tính chất giao hoán: $L_1 \cap L_2 = L_2 \cap L_1$
   - Tính chất kết hợp: $(L_1 \cap L_2) \cap L_3 = L_1 \cap (L_2 \cap L_3)$

- $\forall L_1 : L_1 \cap \varnothing = \varnothing \cap L_1 = \varnothing$ và $L_1 \cap A^* = L_1$
- Tính chất phân phối đối với phép hợp và phép giao:

$$(L_1 \cup L_2) \cap L_3 = (L_1 \cap L_3) \cup (L_2 \cap L_3)$$
$$(L_1 \cap L_2) \cup L_3 = (L_1 \cup L_3) \cap (L_2 \cup L_3)$$

5. Phép lấy phần bù Ngôn ngữ phần bù của ngôn ngữ L trên bảng chữ cái A, kí hiệu $C_A L$, là một ngôn ngữ trên bảng chữ cái A

$$C_A L = \{w \in A^* | w \notin L\}$$

Nhận xét:Phép lấy phần bù có các tính chất sau

- $C_A\{\epsilon\} = A^+, C_A A^+ = \{\epsilon\}$
- $C_A\varnothing = A^*, C_A A^* = \varnothing$
- $C_A\left(C_A L_1 \cup C_A L_2\right) = L_1 \cap L_2$

2.3 Ngôn ngữ chính quy

2.3.1 Định nghĩa

Ngôn ngữ chính quy được định nghĩa như sau:

1. $\varnothing$ là một ngôn ngữ chính quy
2. $\forall \in A, \{a\}$ là ngôn ngữ chính quy, A: bảng chữ cái

3. L_1, L_2 là các ngôn ngữ chính quy thì $L_1 \cup L_2, L_1L_2, L^*$ là các ngôn ngữ chính quy

4. Không có bất kì ngôn ngữ chính quy nào khác ngoài 1,2,3.

2.4 Biểu thức chính quy

2.4.1 Định nghĩa

Một công cụ để biểu diễn ngôn ngữ chính quy là biểu thức chính quy. Biểu được định nghĩa như sau:

1. $\varnothing$ là biểu thức chính quy

2. ϵ là biểu thức chính quy

3. a là ngôn ngữ chính quy với $a \in A$, A là bảng chữ cái hữu hạn

4. r, s là biểu thức chính quy

$$\begin{cases} (r)(s) \to RS \\ r^* \to R^* \qquad \text{là các biểu thức chính quy} \\ (r) + (s) \to R \cup S \end{cases}$$

2.4.2 Các tính chất của biểu thức chính quy

Cho r, s, w là các biểu thức chính quy

1. $r + s = s + r$

2. $(r + s) + w = r + (s + w)$

3. $r\,(s + w) = rs + rw$

4. $r\epsilon = \epsilon r = r$

5. $(\epsilon + r)^* = r^*$

6. $r + \varnothing = r$

7. $(r^*)^* = r^*$

8. $r + r = r$

9. $r\,(sw) = (rs)\,w$

10. $(r + s)\,w = rw + sw$

11. $\varnothing r = r\varnothing = \varnothing$

12. $\varnothing^* = \epsilon$

13. $r + r^* = r^*$

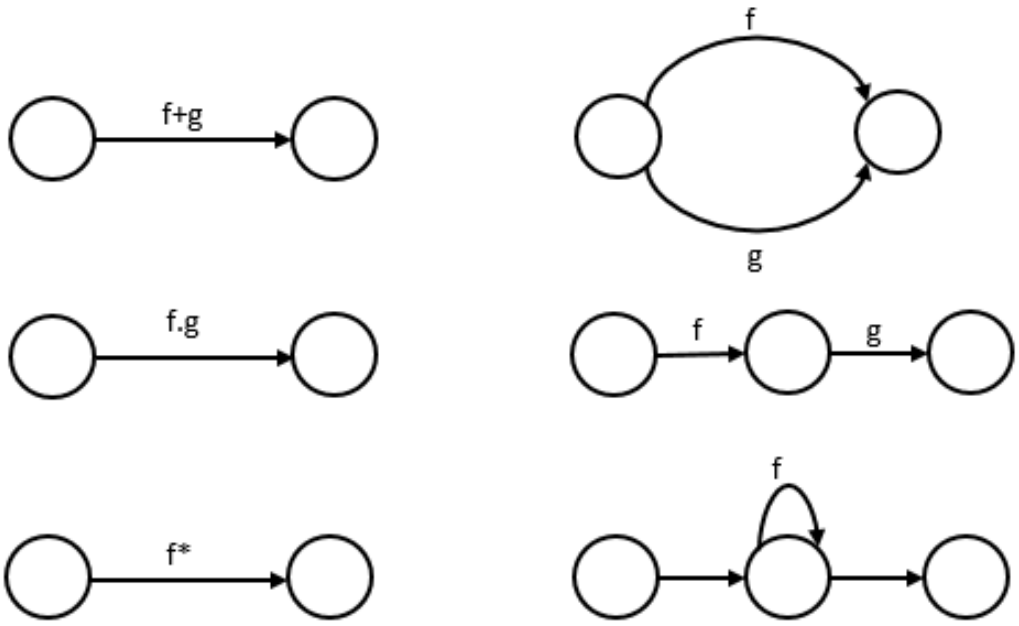
14. $(r^*s^*)^* = (r + s)^*$

15. $(rs)^*r = r(sr)^*$

2.4.3 Đồ thị biểu diễn biểu thức chính quy

- Mỗi đỉnh là một vòng tròn, đỉnh kết thúc được biểu diễn là một vòng tròn nhân đôi

- Đỉnh xuất phát thì mũi tên đi vào
- Các toán tử được biểu diễn như sau:



2.5 Automat hữu hạn

2.5.1 Automat đơn định(DFA)

1. Quy luật để chuyển trạng thái mới cho bởi hàm chuyển:

   $\delta : Q \times A \to Q$

   $\delta(q, a) = p$

   $q, p \in Q, a \in A$

2. Định nghĩa

   M là một DFA:

$$M = (A, Q, \delta, q_0, F)$$

   Trong đó:

   - A: bảng chữ cái hữu hạn
   - Q: tập các trạng thái
   - δ: hàm chuyển
   - q_0: trạng thái bắt đầu, $q_0 \in Q$
   - F: tập trạng thái kết thúc $F \subseteq Q$

2.5.2 Automat đa định(NFA)

1. Quy luật để chuyển trạng thái mới cho bởi hàm chuyển:

   $\delta : Q \times (A \cup \epsilon) \to 2^Q$

   $\delta(q, a) = p$

   $q \in Q, p \in 2^Q, a \in A$

2. Định nghĩa

   M là một DFA:

$$M = (A, Q, \delta, q_0, F)$$

Trong đó:

- A: bảng chữ cái hữu hạn
- Q: tập các trạng thái
- δ: hàm chuyển
- q_0: trạng thái bắt đầu, $q_0 \in Q$
- F: tập trạng thái kết thúc $F \subseteq Q$

Mặc dù trong nhiều trường hợp, ta thấy rằng xây dựng NFA để đoán nhận một ngôn ngữ thường dễ hơn là xây dựng một NFA. Tuy nhiên, người ta đã chứng minh được rằng mọi ngôn ngữ được biểu diễn bởi một NFA nào đó đều có thể biểu diễn được bằng một DFA. Tuy nhiên, trong trượng hợp xấu nhất thì số lượng trạng thái của DFA là 2^n trong khi số lượng trạng thái của NFA là n.

Định lý về sự tương đương. Một ngôn ngữ L được đoán nhận bởi một DFA nào đó khi và chỉ khi nó cũng được đoán nhận bởi một NFA nào đó

2.6 Văn phạm

2.6.1 Định nghĩa

Văn phạm G là một bộ gồm 4 thành phần:

$$G = \{V_T, V_N, S, P\}$$

trong đó:

1. V_T là tập các kí tự kết thúc, mỗi kí tự của nó được gọi là một ký hiệu kết thúc.

2. V_N là tập các kí tự không kết thúc, mỗi phần tử của nó được gọi là một ký hiệu không kết thúc hay ký hiệu phụ

3. $S \in V_N$ được gọi là ký tự xuất phát

4. P là tập hợp các luật sinh có dạng $\alpha \to \beta$, α được gọi là vế trái, còn β được gọi là vế phải của một luật sinh, trong đó α, $\beta \in (V_N \cup V_T)^*$ và α chứa ít nhất một ký tự không kết thúc.

2.6.2 Ngôn ngữ sinh bơi văn phạm

Định nghĩa. Cho văn phạm $G = \{V_T, V_N, S, P\}$ và $\eta, \omega \in (V_N \cup V_T)^*$. Ta nói ω suy dẫn trực tiếp từ η, ký hiệu $\eta \Rightarrow \omega$, nếu tồn tại luật sinh $\alpha \to \beta$, và $\gamma, \sigma \in (V_N \cup V_T)^*$ sao cho $\eta = \gamma\alpha\sigma$, $\omega = \gamma\beta\sigma$

Định nghĩa. Cho văn phạm $G = \{V_T, V_N, S, P\}$ và $\eta, \omega \in (V_N \cup V_T)^*$. Ta nói ω suy dẫn (hay suy dẫn gián tiếp) từ η, ký hiệu $\eta \Rightarrow^* \omega$, nếu $\eta = \omega$ hoặc tồn tại một dãy $D = \omega_0, \omega_1, ...\omega_k$ sao cho $\omega_0 = \eta$, $\omega_k = \omega$ và $\omega_i \Rightarrow \omega_{i+1}$

Định nghĩa. Cho văm phạm $G = \{V_T, V_N, S, P\}$. Xâu $\omega \in V_T^*$ được gọi là sinh bởi văn phạm G nếu tồn tại suy dẫn $S \Rightarrow^* \omega$. Ngôn ngữ sinh bởi văn pham G, ký hiệu là $L(G)$, là tập hợp tất cả các xâu sinh bởi văn phạm G.

$$L(G) = \{\omega \in V_T^* | S \Rightarrow^* \omega\}$$

2.6.3 Văn phạm chính quy

Dựa vào đặc điểm của tập luật sinh mà người ta chia các văn phạm thành các nhóm khác nhau. Noam Chomsky đã phân loại văn phạm thành 4 nhóm:

1. Nhóm 0: Văn phạm không hạn chế
2. Nhóm 1: Văn phạm cảm ngữ cảnh
3. Nhóm 2: Văn phạm phi ngữ cảnh
4. Nhóm 3: Văn phạm chính quy

Trong khuôn khổ của bài báo cáo này, do ta chỉ quan tâm đến ngôn ngữ chính quy, nên bài báo cáo sẽ chỉ trình bày các kết quả liên quan đến văn phạm chính quy.

Định nghĩa. Văn phạm $G = \{V_T, V_N, S, P\}$ mà các quy tắc của nó chỉ có dạng $A \to aB$, $A \to a$ (hoặc chỉ có dạng $A \to Ba$, $A \to a$, trong đó $A, B \in V_N$, $a \in V_T$ được gọi là văn phạm loại 3 hay văn phạm chính quy.
Các văn phạm mà các quy tắc của chúng có dạng trên, đồng thời chưa thêm quy tắc rỗng $S \to \epsilon$ cũng được gọi là văn phạm chính quy.
Các quy tắc trong văn phạm chính quy được gọi là quy tắc chính quy. Ngôn ngữ do văn phạm chính quy suy ra được gọi là ngôn ngữ chính quy.

Chương 3

Tính chất của ngôn ngữ chính quy

3.1 Điều kiện cần và đủ của ngôn ngữ chính quy

3.1.1 Bổ đề bơm

Bổ đề bơm cho ngôn ngữ chính quy. Cho L là một ngôn ngữ chính quy. Khi đó tồn tại n sao cho tất cả các xâu $w \in L$ mà $|w| \geq n$, ta luôn có thể phân tích w thành 3 xâu con, $w = xyz$, thỏa mãn:

1. $y \neq \epsilon$
2. $|xy| \leq n$
3. $\forall\ k \geq 0$, xâu $xy^kz \in L$

Bổ đề này có ý nghĩa rằng ta có thể tìm được một xâu y không quá nhỏ so với w, mà ta có thể "bơm" thêm xâu y vào trong w mà vẫn thu được các xâu thuộc L.

Điều kiện cần và đủ. Cho bảng chữ cái Σ. Ngôn ngữ $L \subset \Sigma*$ là ngôn ngữ chính quy khi và chỉ khi $\exists k \geq 0$ sao cho $\forall y \in \Sigma^*$ thỏa mãn $y = uvw, |v| > 0$ và $\forall z \in \Sigma^*, \forall i \in N$, ta có

- Nếu $yz \in L$ thì $uv^iwz \in L$
- Nếu $yz \notin L$ thì $uv^iwz \notin L$

Chứng minh. ($\Rightarrow$) Do L là ngôn ngữ chính quy, nên tồn tại $M = (Q, \Sigma, \delta, q_0, F)$ là automat hữu hạn đơn định đóan nhận ngôn ngữ L. Gọi n là số trạng thái của M. Chọn $k = n$, khi đó một xâu bất kì có độ dài lơn hơn k đều phải đi qua một trạng thái nào đó ít nhất hai lần. Ta chọn v là xâu ở giữa hai lần liên tiếp đi qua trạng thái đó. Ta dễ dàng có điều phải chứng minh.

($\Leftarrow$) Để chứng minh L là một ngôn ngữ chính quy, ta xây dưng DFA $M = (Q, \Sigma, \delta, q_0, F)$ đoán nhận L như sau. Số trạng thái của M :$|Q| = 1 + |\Sigma| + |\Sigma^2| + ... + |\Sigma^{k-1}|$. Khi chưa quan tâm đến các nút lá, thì M là một cây gốc là q_0 mà trong đó các nút cha có đúng $|\Sigma|$ nút con ứng với cách chuyển trạng thái thì gặp các kí tự trong Σ. Một trạng thái q là trạng thái kết thúc nếu tồn tại một xâu w độ dài $l < k$, $w \in L$ và $\delta(q_0, w) = q$. Để hoàn thiện automat, ta cần phải xây dựng nốt hàm chuyển tại cái nút lá, tức là cần phải gán giá trị cho $\delta(q, \sigma)$ với q là trạng thái ở nút là và $\sigma \in \Sigma$.

Vì automat M hiện tại đang là một cây, nên tại 1 nút q lá chỉ có một đường đi duy nhất từ gốc đến nó. Gọi xâu tương ứng với nút đó là L (l có độ dài $k-1$). Để xác định giá trị của $\delta(q,\sigma)$ ta xem xét xâu $y = l\sigma$. Vì $|y| = k$, nên theo giả thiết tồn tại cách phân tích $y = uvw$ sao cho $\forall z \in \Sigma^*$ nếu $yz \in L$ thì $uv^iwz \in L$ và ngược lại. Cho $\delta(q,\sigma) = \delta(q_0, uw)$. Từ đó, ta dễ dàng chứng minh được M là automat đoán nhận ngôn ngữ L

3.1.2 Ứng dụng của bổ đề bơm

Trong phần này, bài báo cáo sẽ trình bày một vài ví dụ mà ở đó bổ đề bơm được sử dụng để chúng minh rằng một ngôn ngữ không là ngồn ngữ chính quy.

Ví dụ 1. Chứng minh răng ngôn ngữ L_{eq} gồm các xâu nhị phân có số chữ số 0 và số số chữ số 1 bằng nhau không là một ngôn ngữ chính quy.
Giả sử phản chứng rằng L_{eq} là ngôn ngữ chính quy. Khi đó theo bổ để bơm tồn tại n thỏa mãn 3 tính chất nêu ở trên. Ta xét $w = 0^n1^n$, dễ thấy rằng $w \in L_{eq}$ do số chữ số 0 bằng số chữ số 1 và bằng n. Theo tính chất 2 của bổ đề bơm, ta phân tích $w = xyz$ và $|xy| \leq n$. Ta thấy rằng xy là prefix của w với đồ dài nhỏ hơn n, do đó xâu xy gồm toàn chữ số 0. Theo tính chất 3 của bổ đề bơm, ta có $xz \in L$ nhưng vì y là xâu khác xâu rỗng nên y chứa ít nhất một số 0. Suy ra xz sẽ có số số 0 ít hơn số số 1 (mẫu thuẫn). Vậy L_{eq} không là một ngôn ngữ chính quy.

Ví dụ 2. Chứng minh rằng ngôn ngữ L_{pr} gồm các xâu nhị phân chứa một số nguyên tố số 1 không là một ngôn ngữ chính quy.
Giả sử rằng L là ngôn ngữ chính quy. Gọi k là số nguyên tố lơn hơn n. Ta chọn xâu nhị phân w gồm k chữ số 1. Theo định nghĩa, ta có w thuộc vào ngôn ngữ L_{pr}. Theo bổ đề bơm, ta có $w = xyz$, $y \neq \epsilon$ và $xy^iz \in L \forall k \geq 0$. Gọi độ dài của xâu y là m. Ta có $k + im$ là số nguyên tố với mọi i (vô lý). Vậy L_{pr} không là một ngôn ngữ chính quy.

3.2 Tính đóng của ngôn ngữ chính quy

Từ định nghĩa, ta thấy rằng ngôn ngữ chính quy là đóng với phép toán hợp, tích ghép và phép toán sao. Tuy nhiên, ngôn ngữ chính quy còn đóng đối với các phép toán khác, được liệt kê và chứng minh dưới đây.

1. Hợp của hai ngôn ngữ chính quy là ngôn ngữ chính quy
2. Giao của hai ngôn ngữ chính quy là ngôn ngữ chính quy
3. Phần bù của một ngôn ngữ chính quy là một ngôn ngữ chính quy
4. Phần khác giữa hai ngôn ngữ chính quy là ngôn ngữ chính
5. Nghịch đảo của ngôn ngữ chính quy là một ngôn ngữ chính quy
6. Phép toán sao trên một ngôn ngữ chính quy tạo ra một ngôn ngữ chính quy
7. Tích ghép của hai ngôn ngữ chính quy là một ngôn ngữ chính quy

Chú ý rằng khi thực hiện các phép toán trên hai ngôn ngữ xây dựng trên hai bảng chữ cái khác nhau, ta hoàn toàn có thể xây dựng bảng chữ cái mới chung cho cả hai ngôn ngữ bằng việc lấy hợp của hai bảng chữ cái ban đầu. Do đó, để tránh việc giải thích nhiều lần, khi thực hiện phép toán trên hai ngôn ngữ, ta có thể giả sử rằng hai ngôn ngữ đó được xây dựng trên cùng một bảng chữ cái.

Định lý. Nếu L là ngôn ngữ chính quy trên bảng chữ cái Σ, thì $\overline{L} = \Sigma^* - L$ cũng là ngôn ngữ chính quy.

Chứng minh. Do là một ngôn ngữ chính quy, gọi $L = L(A)$ trong đó A là 1 DFA gồm năm thành phần $(Q, \Sigma, \delta, q_0, F)$ (trong đó hàm chuyển δ là hàm chuyển đầy đủ). Khi đó $\overline{L} = L(B)$ với DFA $B = (Q, \Sigma, \delta, q_0, Q - F)$. Tức là B giống với A chỉ khác rằng trạng thái chấp nhận được của B là trạng thái không chấp nhận được của A, và ngược lại. Khi đó một xâu $w \in L(B)$ khi và chỉ khi $\delta(q_0, w) \in Q - F$, điều này chỉ xảy ra khi và chỉ khi $w \notin L(A)$

Ba phép toán trên tập hợp (hợp, giao, và lấy phần bù) không hoàn toàn độc lập với nhau. Do đó ta có thể dễ dàng chứng minh được giao của hai ngôn ngữ chính quy là một ngôn chính quy dựa vào công thức sau

$$L \cap M = \overline{\overline{L} \cup \overline{M}}$$

Tương tự như vậy ta có thể chứng minh hiệu (hay phần khác) giữa hai

ngôn ngữ chính quy là một ngôn ngữ chính quy dừa vào công thức

$$L - M = L \cap \overline{M}$$

Định lý. Nếu L là ngôn ngữ chính quy, thì L^R cũng là ngôn ngữ chính quy

Chứng minh. Giả sử L là ngôn ngữ chính quy được biểu diễn bởi biểu thức chinh quy E. Ta sẽ chứng minh định lý trên bằng quy nạp theo kích cỡ của E. Ta cần chỉ ra rằng tồn tại một biểu thức chính quy E^R sao cho $L(E^R) = (L(E))^R$. Thật vậy:

Nếu E là rỗng, kí tự rỗng hoặc một kí tự a thuộc bảng chữ cái Σ, thì ta dễ thấy rằng $E^R = E$ là biểu thức chính quy ta cần tìm.

Ta chứng minh quy nạp theo 3 trường hợp sau:

1. Nếu $E = E_1 + E_2$ thì khi đó chọn $E^R = E_1^R + E_2^R$
2. Nếu $E = E_1E_2$ thì $E^R = E_2^R E_1^R$
3. Nếu $E = E_1^*$ thì $E^R = (E_1^R)^*$

Bằng sử dụng tính chất của biểu thức chính quy được liệt kê ở chương 1, ta có thể chứng minh được nếu E_1^R, E_2^R thỏa mãn $L(E_1^R) = (L(E_1))^R$ và $L(E_2^R) = (L(E_2))^R$ thì $L(E^R) = (L(E))^R$. Như vậy bằng quy nạp, ta có điều phải chứng minh.

3.3 Tính quyết định của ngôn ngữ chính quy

Trong phần này, ta xem xét những thuật toán để trả lời cho các câu hỏi liên quan đến ngôn ngữ chính quy:

1. Ngôn ngữ được miêu tả có rỗng hay không ?
2. Một xâu w cho trước có thuộc một ngôn ngữ hai không
3. Hai ngôn ngữ được miêu tả có biểu diễn cùng một ngôn ngữ hay không ?

Trong nhiều trường hợp, ngôn ngữ chúng ta đang xem xét không được cho dưới dạng tường minh như một tập hợp, mà được miêu tả bằng một automat hữu hạn. Khi đó việc kiểm tra ngôn ngữ rỗng không còn hiển nhiên nữa. Tuy nhiên bài toán này tương đương với tìm đường đi trên đô thị bắt đầu từ đỉnh xuất phát và kết thúc là đỉnh chấp nhận được.
Việc kiểm tra một xâu cho trước có thuộc một ngôn ngữ hay không là đơn giản nếu ta xây dựng được DFA hoặc NFA đoán nhận ngôn ngữ đó.
Cuối cùng, việc chuyển đổi giữa các cách biểu diễn một ngôn ngữ chính quy được miêu tả chi tiết trong Chương 4.

Chương 4

Thuật Toán

Algorithm 1 Chuyển từ NFA sang DFA

Input (NFA) $M = \{Q, A, \delta, q_0, F\}$

Output (DFA) $M' = \{Q', A, \delta', q_0', F'\}$

Khởi tạo $Q' = \{q_0\}, F' = \emptyset$

while $\exists \delta'(q_x, a)$ chưa xác định với $q_x \in Q', a \in A$ **do**

Chọn $q_x' \in Q', a \in A$ sao cho $\delta'(q_x', a)$ chưa xác định

Gán $q_{xa} = \delta'(q_x, a) = \{q \in Q | \delta(q', a) = q, q' \in q_x'\}$

if $q_{xa} \notin Q'$ **then**

$Q' = Q' \cup q_{xa}$

end if

if $q_{xa} \cap F \neq \emptyset$ **then**

$F' = F' \cup q_{xa}$

end if

end while

Bổ đề Arden: Với X, A, B là các xâu kí tự, phương trình $X = A.X + B$ có nghiệm duy nhất là $X = A^*.B$

Xét DFA hoặc NFA $M = \{Q, A, \delta, q_0, F\}$ có n trạng thái, với mỗi trạng thái q_i ta đưa ra một phương trình có dạng:

$$Q_i = \bigcup_{q_i \xrightarrow{a} q_j} a.Q_j + \begin{cases} \{\varepsilon\} & , \ q_i \in F \\ \emptyset & , \text{ngược lại} \end{cases}$$

Ta thấy ta luôn có thể đưa về hệ phương trình như sau:

$$X_i = B_i + A_{i,1}X_1 + \ldots + A_{i,n}X_n \text{ trong đó } i = 1, 2, ..., n$$

Ta có thể giải hệ này để tìm ra được biểu thức chính quy tương ứng.

Xét với $i = n$:

$$X_n = B_n + A_{n,1}X_1 + \ldots + A_{n,n}X_n$$

Theo bổ đề Arden, phương trình trên sẽ có nghiệm là

$$X_n = A^*_{n,n}(B_n + A_{n,1}X_1 + \ldots + A_{n,n-1}X_{n-1})$$

Bằng cách đặt $B'_n = A^*_{n,n}B_n$ và $A'_{n,i} = A^*_{n,n}A_{n,i}$, ta được:

$$X_n = B'_n + A'_{n,1}X_1 + \ldots + A'_{n,n-1}X_{n-1}$$

Do đó, ta có thể bỏ X_n khỏi các phương trình của hệ bằng cách đặt:

$$B'_i = B_i + A_{i,n}B'_n$$

$$A'_{i,j} = A_{i,j} + A_{i,n}A'_{n,j}$$

với $i, j < n$.

Làm như vậy cho đến khi $i = 1$, ta sẽ được phương trình có dạng:

$$X_1 = B'_1$$

Đây chính là biểu thức chính quy cần tìm.

Algorithm 2 Tìm ngôn ngữ chính quy được đón nhận bởi FA

Input DFA hoặc NFA $M = \{Q, A, \delta, q_0, F\}$

Output Biểu thức chính quy biểu diễn ngôn ngữ chính quy tương ứng

Khởi tạo mảng B gồm n phần tử, $B[i] = \epsilon$ nếu $q_i \in F$, ngược lại $B[i] = \emptyset$.

Khởi tạo mảng A kích thước $n \times n$, $A[i, j] = \emptyset$ với mọi $i, j \in \{0, n-1\}$

Nếu $\delta q_i, a = q_j$ thì đặt $A[i, j] = a$

for $m = n$ to 1 **do**

  Gán $B[m] = (A[m, m])^*.B[m]$

  for $j = 1$ to m **do**

    Gán $A[m, j] = (A[m, m])^*.A[m, j]$

  end for

  for $i = 1$ to m **do**

    Gán $B[i] = B[i] + A[i, m].B[m]$

    for $j = 1$ to m **do**

      Gán $A[i, j] = A[i, j] + A[i, m].A[m, j]$

    end for

  end for

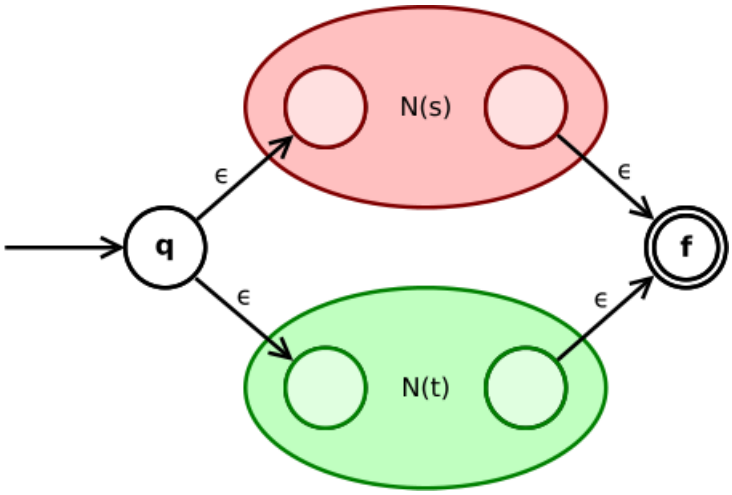
end for

Ta sẽ xây dựng NFA như sau:

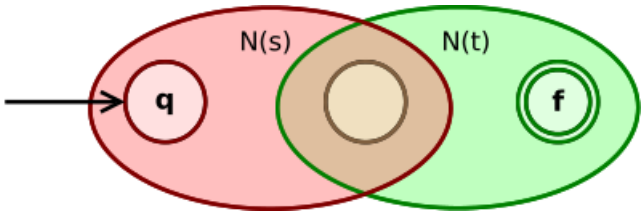
- Kí tự rỗng:

- Kí tự $a \in A$:

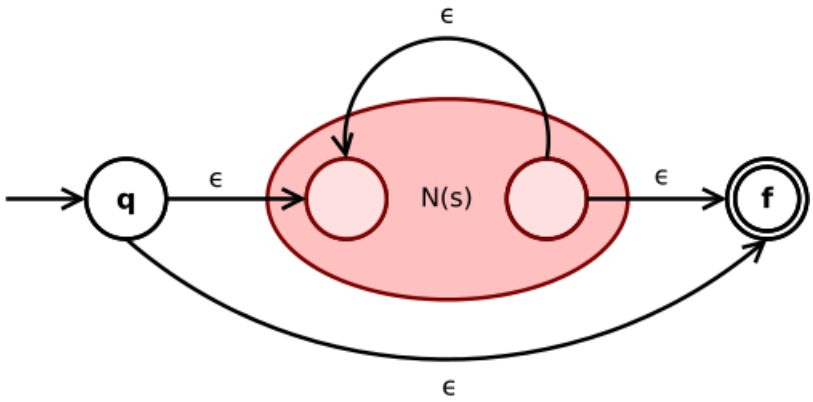
- Phép hợp $s + t$:



- Phép tích ghép $s.t$:



- Phép s^*:



Algorithm 3 Xây dựng FA đoán nhận ngôn ngữ chính quy

Input Biểu thức chính quy s biểu diễn ngôn ngữ chính quy

Output FA $M = \{Q, A, \delta, q_0, F\}$

Duyệt biểu thức chính quy, thực hiện xây dựng NFA như mô tả bên trên. Từ NFA trên ta tiếp tục thực hiện chuyển từ NFA sang DFA, tối tiểu hóa nếu cần.

Algorithm 4 Chuyển từ văn phạm tuyến tính trái sang văn phạm tuyến tính phải

Input Văn phạm tuyến tính trái $G_T = \{V_T, V_N, S, P\}$

Output Văn phạm tuyến tính phải $G_P = \{V'_T, V'_N, S', P'\}$

if tồn tại luật có dạng $S \to Sw$ với $w \in V_T*$ **then**

Thêm vào V_N S_0, thêm luật $S_0 \to S$ vào P, đặt $S = S_0$.

end if

Khởi tạo $V'_T = V_T, V'_N = V_N, P' = \emptyset, S' = S$

for each luật $p \in P$ **do**

if luật có dạng $S \to p$ **then**

Thêm luật đó vào P'

else if luật có dạng $A \to w$ **then**

Thêm luật $S \to wA$ vào P'

else if luật có dạng $B \to Aw$ **then**

Thêm luật $A \to wB$ vào P'

else if luật có dạng $S \to Aw$ **then**

Thêm luật $A \to w$ vào P'

end if

end for

Algorithm 5 Xây dựng NFA từ văn phạm chính quy

Input Văn phạm chính quy $G = \{V_T, V_N, S, P\}$

Output NFA $M = \{Q, A, \delta, q_0, F\}$ tương ứng với văn phạm đã cho.

if Văn phạm G là tuyến tính trái **then**

Chuyển G về dạng tuyến tính phải

end if

Khởi tạo $A = V_T$

for each $v \in V_N$ **do**

Tạo trạng thái q_v mới tương ứng.

Thêm q_v vào Q.

end for

Gán $q_0 = q_S$

Thêm trạng thái q_ϵ vào Q, đặt $F = \{q_\epsilon\}$

for each luật $p \in P$ **do**

Với $p = A \to wB$ trong đó $A \in V_N, B \in V_N \cup \{\epsilon\}, w \in V_T^*, w = w_1.w_2...w_n$

Thêm các trạng thái $q_{pw_1}, q_{pw_2}, ..., q_{pw_{n-1}}$ vào Q.

Gán $\delta(q_{pw_i}, w_{i+1}) = q_{pw_{i+1}}$ với $i = 1, ..., n-2$.

Gán $\delta(q_A, w_1) = q_{pw_1}$, $\delta(q_{pw_{n-1}}, w_n) = q_B$.

end for

Algorithm 6 Chuyển từ NFA sang văn phạm tuyến tính phải

Input NFA $M = \{Q, A, \delta, q_0, F\}$

Output Văn phạm tuyến tính phải $G = \{V_T, V_N, S, P\}$ tương ứng với NFA đã cho.

Khởi tạo $V_T = A, V_N = Q, S = q_0$

for each $q_i \in Q$ **do**

  for each $a \in A$ **do**

    for each $q_j \in \delta(q_i, a)$ **do**

      Thêm luật $q_i \to aq_j$ vào P

    end for

  end for

end for

for each $q_i \in F$ **do**

  Thêm luật $q_i \to \epsilon$ vào P

end for

Algorithm 7 Tìm ngôn ngữ chính quy được sinh bởi văn phạm chính quy

Input Văn phạm chính quy $G = \{V_T, V_N, S, P\}$

Output Ngôn ngữ chính quy ứng với văn phạm đã cho

Chuyển văn phạm chính quy về NFA

Từ NFA thực hiện chuyển về ngôn ngữ chính quy.

Algorithm 8 Tìm ngôn ngữ chính quy sinh ra bởi văn phạm tuyến tính phải

Input Ngôn ngữ chính quy L

Output Văn phạm chính quy $G = \{V_T, V_N, S, P\}$ ứng với biểu thức chính quy đã cho

Tìm NFA đoán nhận ngôn ngữ chính quy L

Từ NFA thực hiện chuyển về văn phạm tuyến tính phải.

Chương 5

Sơ lược về lý thuyết mã

5.1 Mã

A là bảng chữ. X được gọi là mã nếu $\forall m, n$ và $x_1, x_2, ..., x_n, y_1, y_2, ..., y_m \in X$ thỏa mãn điều kiện:

$$x_1x_2...x_n = y_1y_2...y_m$$

khi đó, $m = n$ và $x_i = y_i, i = \overline{1, n}$

Hay nói cách khác, một tập X là mã nếu tất cả các từ trong X^+ đều chỉ có một cách phân tích duy nhất thành các từ trong X. Từ định nghĩa trên ta dễ thấy tập con của một mã cũng là một mã, và mã không thể chứa kí tự ϵ

5.1.1 Tính chất của mã

Định nghĩa về mã còn có thể phát biểu bằng một cách khác như sau

Bổ đề 1.1 Cho X là tập con của A^* là một mã, khi đó nếu một đồng cấu $\beta : B^* \to A^*$ chứa một song anh đi từ B vào X thì nó là đơn ánh. Ngược lại nếu tồn tại một đồng cấu đơn ánh $\beta : B^* \to A^*$ sao cho $X = \beta(B)$ thì X là một mã.

Bổ đề 1.1 là lý do tại sao một tập X được gọi là mã. Bản chất của việc mã hóa là tương ứng một kí tự $b \in B^*$ với một kí tự $a \in A^*$ theo một quy tắc mã hóa β nào đó. Việc β là đơn ánh đảm bảo rằng việc giải mã các xâu nhận được từ việc mã hóa là duy nhất.

Hệ quả 1.2. Cho $\alpha : A^* \to C^*$ là một đồng cấu và đơn ánh. Khi đó, nếu X là một mã trên A thì $\alpha(X)$ là một mã trên C. Ngược lại, ta cũng có, nếu Y là một mã trên C, thì $\alpha^{-1}(Y)$ cũng là một mà trên A

Hệ quả 1.3. Nếu $X \subset A^*$ là một mã, thì X^n cũng là một mã trện $A \; \forall \; n > 0$

5.1.2 Các loại mã đặc biệt

Cho X là một tập con của A^*. Khi đó, X được gọi là tập prefix nếu không phần tử nào trong X là prefix của phần tử khác trong X. Nếu ta xây dựng một quan hệ thứ tự prefix $x \leq x'$ nếu x là prefix của x', thi khi đó X là một mã nếu mọi phần tử x, x' trong X mà $x \leq x'$ thì $x = x'$. Hay nói cách

khác, X là một mã nếu hai phần tử phân biệt bất kì trong X đều không thể so sánh được với nhau theo quan hệ thứ tự prefix.

Từ định nghĩa trên, ta dễ thấy rằng nếu một tập prefix chứa kí tự ϵ thì tập đó chỉ chứa duy nhất một kí tự là ϵ.

Tập suffice được định nghĩa bằng một cách tương tự: Tập suffix là tập mọi phần tử đều không là suffix của các phần tử khác trong nó. Một tập là biprefix khi nó vừa là tập prefix, vừa là tập suffix.

Bổ đề 1.4 Mọi tập prefix (suffix, biprefix) khác $\{\epsilon\}$ đều là một mã. **Chứng minh.** Giả sử rằng X không phải là mã, khi đó tồn tại ít nhất một xâu w có hai cách phân tích khác nhau. Gọi xâu có độ dài ngắn nhất như vậy là w

$$w = x_1x_2...x_n = x'_1x'_2...x'_m$$

Vì x_1, x'_1 khác kí tự rỗng, và w là xâu có độ dài ngắn nhất nên $x_1 \neq x'_1$. Khi đó $x_1 < x'_1$ hoặc $x'_1 < x_1$, mâu thuẫn với giả thiết của tập prefix. Vậy tập prefix là mã. Ta cũng có thể có chứng minh tương tự đối với tập suffix.

5.1.3 Mã tối đại

Mã X được gọi là tối đại (theo nghĩa bao hàm) trên A nếu X không là con thực sự trong một mã nào khác trên A.

Chú ý rằng tính tối đại của một mã phụ thuộc vào bảng chữ cái mà nó được xây dựng lên. Nếu $X \subset A^*$ là một mã tối đại trên A và A là tập con thực sự của B thì khi đó X không đảm bảo là mã tối đại trên B.

Bổ đề 1.5 Cho A là một bảng chữ cái. Khi đó mọi ví nhóm con M thuộc

A^* đều có duy nhất một tập sinh cực tiểu $X = (M-1) - (M-1)^2$

Bổ đề 1.6 Nếu M là một vị nhóm tự do con của A^*, khi đó tập sinh cực tiểu của M là một mã. Ngược lại, nếu $X \subset A^*$ là một mã, thì vị nhóm con X^* là vị nhóm tự do và X là tập sinh cực tiểu của nó.

Bổ đề 1.7 Nếu M là vị nhóm con tự do tối đại của A^* thì cơ sở X của nó là mã tối đại.

5.2 Tiêu chuẩn kiểm định mã

Bài toán kiểm định một tập cho trước là một mã hay không là bài toán khó. Trong phần này, thuật toán được miêu tả không sử dụng các tính chất đặc biệt của mã mà chỉ sử dụng các phép tính toán hợp lý để kiểm định mã và thuật toán chỉ có thể thực hiện được với mã hữu hạn.

Thuật toán được trình bày dưới đây tính tất cả các phần dư của mọi cách thử phân tích xâu thành hai cách khác nhau. Thuật toán phát hiện được một xâu tồn tại nhiều hơn 2 cách phân tích khác nhau khi phần dư chứa kí tự rỗng ϵ

Một cách cự thể, các phần dư được thực hiện lần lượt như sau. Gọi X là một tập con của A^*, và

$$U_1 = X^{-1}X - \epsilon$$

$$U_{n+1} = X^{-1}U_n \cup U_n^{-1}X$$

Trong đó $N^{-1}D = \{y | xy \in D \ v \ x \in N\}$

Khi đó ta có kết quả sau:

Định lý 2.1 Tập $X \subset A^*$ là một mã khi và chỉ khi các tập U_n không chứa

kí tự rỗng (với mọi n).

Bổ đề 2.2 Nếu X là tập một tập hữu hạn thì tập các U_n là một tập hữu hạn

Chú ý rằng ta dễ có kết quả của bổ đề 2.2 vì U_n chỉ gồm suffix của những xâu trong X

Từ hai kết quả trên, ta có thể miêu tả thuật toán kiểm định mã cho một tập hữu hạn X như sau Bài toán:

- Input: Một tập X gồm hữu hạn các xâu thuộc A^+
- Output: X có phải phải là mã hay không ?

1. Tính $S_1 = X^{-1}X \setminus \{\epsilon\}$
2. Thực hiện vòng lặp tính và lưu lại giá trị của U_{i+1} theo công thức

$$U_{n+1} = X^{-1}U_n \cup U_n^{-1} \; vi \; i \geq 1$$

3. Nếu trong quá trình tính toán, tại một giá trị i nào đó mà U_n chứa kí tự rỗng thì dừng thuật toán và kết luận X không phải là mã.
4. Nếu trong quá trình tính toán, tìm được U_j mà $U_j = U_i$ với $i < j$ thì dừng thuật toán và kết luận X là mã

Đính lý 2.1 và bổ đề 2.2 giúp khẳng định rằng thuật toán trên sẽ dừng sau một số hữu hạn bước tính toán.

Chương 6

Ứng dụng của lý thuyết mã

6.1 Mã Hamming

6.1.1 Giới thiệu

Mã Hamming là một mã sửa lỗi, được đặt tên theo người phát minh ra nó, Richard Hamming. Mã Hamming có thể phát hiện một bit hoặc hai bit lỗi. Mã Hamming còn có thể sửa các lỗi do một bit bị sai gây ra.

6.1.2 Lịch sử phát triển

Mã chẵn lẻ

Trước mã hamming, người ta sử dụng mã chẵn lẻ. Mã chẵn lẻ thêm một bit vào trong dữ liệu, và bit này cho biết số lượng bit có giá trị 1 của đoạn dữ liệu nằm trước là một số chẵn hay một số lẻ. Nếu một bit bị thay đổi trong quá trình truyền dữ liệu, giá trị chẵn lẻ trong thông điệp sẽ thay đổi và do đó có thể phát hiện được lỗi. Theo quy ước chung, bit kiểm tra có giá trị bằng 1 nếu số lượng bit có giá trị một trong dữ liệu là một số lẻ, và giá trị của bit kiểm tra bằng 0 nếu số lượng bit có giá trị một trong dữ liệu là một số chẵn. Nói cách khác, nếu đoạn dữ liệu và bit kiểm tra được gộp lại cùng với nhau, số lượng bit có giá trị bằng 1 luôn luôn là một số chẵn.

Việc kiểm tra dùng mã chẵn lẻ là một việc không được chắc chắn cho lắm, vì nếu số bit bị thay đổi là một số chẵn thì mã này không phát hiện được lỗi. Hơn nữa, mã chẵn lẻ không biết đươck bit nào là bit lỗi, kể cả khi nó phát hiện có lỗi xảy ra. Toàn bộ dữ liệu đã nhận được phải bỏ đi và truyền lại từ đầu.

Mã hai-trong-năm

Trong những năm của thập niên kỷ 1940, Bell có sử dụng một mã hiệu phức tạp hơn một chút, gọi là mã hai-trong-năm (two-out-of-five code). Mã này đảm bảo mỗi một khối 5 bit (còn được gọi là khối-5) có chính xác hai bit có giá trị bằng 1. Máy tính có thể nhận ra là dữ liệu nhập vào có

lỗi nếu trong một khối 5 bit không 2 bit có giá trị bằng 1. Tuy thế, mã hai-trong-năm cũng chỉ có thể phát hiện được một đơn vị bit mà thôi; nếu trong cùng một khối, một bit bị lộn ngược thành giá trị 1, và một bit khác bị lộn ngược thành giá trị 0, quy luật hai-trong-năm vẫn cho một giá trị đúng (remained true), và do đó nó không phát hiện là có lỗi xảy ra.

Tái diễn dữ liệu

Một mã nữa được dùng trong thời gian này là mã hoạt động bằng cách nhắc đi nhắc lại bit dữ liệu vài lần (tái diễn bit được truyền) để đảm bảo bit dữ liệu được truyền, truyền đến nơi nhận trọn vẹn. Chẳng hạn, nếu bit dữ liệu cần được truyền có giá trị bằng 1, một mã tái diễn $n = 3$ sẽ cho truyền gửi giá trị "111". Nếu ba bit nhận được không giống nhau, thì hiện trạng này báo cho ta biết rằng, lỗi trong truyền thông đã xảy ra. Nếu kênh truyền không bị nhiễu, tương đối đảm bảo, thì với hầu hết các lần truyền, trong nhóm ba bit được gửi, chỉ có một bit là bị thay đổi. Do đó các nhóm $001, 010$, và 100 đều tương đương cho một bit có giá trị 0, và các nhóm 110, 101, và 011 đều tương đương cho một bit có giá trị 1 - lưu ý số lượng bit có giá trị 0 trong các nhóm được coi là có giá trị 0, là đa số so với tổng số bit trong nhóm, hay 2 trong 3 bit, tương đương như vậy, các nhóm được coi là giá trị 1 có số lượng bit bằng 1 nhiều hơn là các bit có giá trị 0 trong nhóm - chẳng khác gì việc các nhóm bit được đối xử như là "các phiếu bầu" cho bit dữ liệu gốc vậy. Một mã có khả năng tái dựng lại thông điệp gốc trong một môi trường nhiễu lỗi được gọi là mã "sửa lỗi" (error-correcting code).

Tuy nhiên, những mã này không thể sửa tất cả các lỗi một cách đúng

đắn hoàn toàn. Chẳng hạn chúng ta có một ví dụ sau: nếu một kênh truyền đảo ngược hai bit và do đó máy nhận thu được giá trị "001", hệ thống máy sẽ phát hiện là có lỗi xảy ra, song lại kết luận rằng bit dữ liệu gốc là bit có giá trị bằng 0. Đây là một kết luận sai lầm. Nếu chúng ta tăng số lần các bit được nhắc lại lên 4 lần, chúng ta có thể phát hiện tất cả các trường hợp khi 2 bit bị lỗi, song chúng ta không thể sửa chữa chúng được (số phiếu bầu "hòa"); với số lần nhắc lại là 5 lần, chúng ta có thể sửa chữa tất cả các trường hợp 2 bit bị lỗi, song không thể phát hiện ra các trường hợp 3 bit bị lỗi.

Nói chung, mã tái diễn là một mã hết sức không hiệu quả, giảm công suất xuống 3 lần so với trường hợp đầu tiên trong ví dụ trên của chúng ta, và công suất làm việc giảm xuống một cách nhanh chóng nếu chúng ta tăng số lần các bit được nhắc lại với mục đích để sửa nhiều lỗi hơn.

6.1.3 Xây dựng mã Hamming

Hamming nghiên cứu các kế hoạch mã hóa hiện có, rồi tổng quát hóa khái niệm. Ông xây dựng một danh mục để diễn tả hệ thống máy, bao gồm cả số lượng bit dùng cho dữ liệu và các bit sửa lỗi trong một khối. Chẳng hạn, nếu một từ dữ liệu có 8 bit chỉ gồm 7 bit chứa dữ liệu và phải thêm 1 bit sửa lỗi vào, ông diễn tả phương pháp này là mã $(8, 7)$. Như vậy mã tái diễn sẽ được gọi là mã $(3, 1)$. Ông đưa ra khái niệm "tỷ lệ thông tin", được tính bằng lấy con số thứ hai chia cho con số thứ nhất. Ví dụ, với mã tái diễn, tỷ lệ thông tin sẽ là $\frac{1}{3}$.

Hamming còn phát hiện ra nan đề với việc đảo giá trị của hai hoặc hơn hai bit nữa, và miêu tả nó là "khoảng cách" (distance) (hiện nay nó được gọi là khoảng cách Hamming (Hamming distance) - theo cái tên của ông). Mã chẵn lẻ có khoảng cách bằng 2, vì nếu có 2 bit bị đảo ngược thì lỗi trong truyền thông trở nên vô hình, không phát hiện được. Mã tái diễn $(3, 1)$ có khoảng cách là 3, vì 3 bit, trong cùng một bộ ba, phải bị đổi ngược trước khi chúng ta được một từ mã khác. Mã tái diễn $(4, 1)$ (mỗi bit được nhắc lại 4 lần) có khoảng cách bằng 4, nên nếu 2 bit trong cùng một nhóm bị đảo ngược thì lỗi đảo ngược này sẽ đi thoát mà không bị phát hiện.

Cùng một lúc, Hamming quan tâm đến hai vấn đề; tăng khoảng cách và đồng thời tăng tỷ lệ thông tin lên, càng nhiều càng tốt. Trong những năm thuộc niên kỷ 1940, ông đã xây dựng một số kế hoạch mã hóa. Những kế hoạch này đều dựa trên những mã đang tồn tại song được nâng cấp và tiến bộ một cách sâu sắc. Bí quyết chìa khóa cho tất cả các hệ thống của ông là việc cho các bit chẵn lẻ gối lên nhau (overlap), sao cho chúng có khả năng tự kiểm tra lẫn nhau trong khi cùng kiểm tra được dữ liệu nữa.

Thuật toán cho việc xây dựng mã Hamming như sau:

- Tất cả các bit ở vị trí là các số mũ của 2 được dùng làm bit chẵn lẻ. (các vị trí như 1, 2, 4, 8, ...)
- Tất cả các vị trí bit khác được dùng cho dữ liệu sẽ được mã hóa.
- Mỗi bit chẵn lẻ đếm số lượng bit 1 cho một số bit trong từ mã. Vị trí

của bit chẵn lẻ quyết định chuỗi các bit mà nó kiểm tra.

- Bit chẵn lẻ ở vị trí 1 kiểm tra các bit có 1 ở cuối trong vị trí được viết dưới dạng nhị phân (1=1, 3=11, 5=101, 7=111, ...)
- Bit chẵn lẻ ở vị trí 2 kiểm tra các bit có 1 ở thứ hai trong vị trí được viết dưới dạng nhị phân (2=10, 3=11, 6=110, 7=111, ...)
- Bit chẵn lẻ ở vị trí 4 kiểm tra các bit có 1 ở thứ ba trong vị trí được viết dưới dạng nhị phân (4-7, 12-15, 20-23, ...)
- Bit chẵn lẻ ở vị trí 8 kiểm tra các bit có 1 ở thứ tư trong vị trí được viết dưới dạng nhị phân (8-15, 24-31, 40-47, ...)

6.1.4 Ví dụ mã Hamming (11, 7)

Lấy ví dụ chúng ta có một từ dữ liệu dài 7 bit với giá trị là "0110101". Để chứng minh phương pháp các mã Hamming được tính toán và được sử dụng để kiểm tra lỗi, xin xem bảng liệt kê dưới đây. Chữ d (data) được dùng để biểu thị các bit dữ liệu và chữ p (parity) để biểu thị các bit chẵn lẻ (parity bits).

Thứ tự bit	1	2	3	4	5	6	7	8	9	10	11
Vị trí bit chẵn lẻ và các bit dữ liệu	p_1	p_2	d_1	p_4	d_2	d_3	d_4	p_8	d_5	d_6	d_7
Nhóm dữ liệu (không có bit chẵn lẻ):			**0**		**1**	**1**	**0**		**1**	**0**	**1**
p_1	1		0		1		0		1		1
p_2		**0**	0			1	0			0	1
p4				**0**	1	1	0				
p8								**0**	1	0	1
Nhóm dữ liệu (với bit chẵn lẻ):	1	**0**	0	**0**	1	1	0	**0**	1	0	1

Cách tính các bit chẵn lẻ trong mã Hamming (từ trái sang phải)

Dữ liệu mới - bao gồm các bit chẵn lẻ - bây giờ là "10001100101". Nếu chúng ta thử cho rằng bit cuối cùng bị sai từ 1 sang 0. Nhóm dữ liệu mới sẽ là "10001100100"; Dưới đây, chúng ta sẽ phân tích quy luật kiến tạo mã Hamming bằng cách cho bit chẵn lẻ giá trị 1 khi kết quả kiểm tra dùng quy luật số chẵn bị sai.

Thứ tự bit	1	2	3	4	5	6	7	8	9	10	11		
Vị trí bit chẵn lẻ và các bit dữ liệu	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7	Kiểm chẵn lẻ	Bit chẵn lẻ
Nhóm dữ liệu nhận được:	1	0	0	0	1	1	0	0	1	0	0	1	
p_1	1		0		1		0		1		0	Sai	1
p_2		0	0			1	0			0	0	Sai	1
p4				0	1	1	0					Đúng	0
p8								0	1	0	0	Sai	1

Kiểm tra các bit chẵn lẻ (bit bị đảo lộn có nền thẫm)

Bước cuối cùng là định giá trị của các bit chẵn lẻ (nên nhớ bit nằm dưới cùng được viết về bên phải - viết ngược lại từ dưới lên trên). Giá trị số nguyên của các bit chẵn lẻ là 11, và như vậy có nghĩa là bit thứ 11 trong nhóm dữ liệu (data word) - bao gồm cả các bit chẵn lẻ - là bit có giá trị không đúng, và bit này cần phải đổi ngược lại.

	p8	p4	p_2	p_1	
Nhị phân	1	0	1	1	
Thập phân	8		2	1	Σ = 11

Bằng việc sửa lỗi và bỏ đi phần mã Hamming, chúng ta lấy được phần dữ liệu gốc với giá trị là 0110101.

Lưu ý, các bit chẵn lẻ không kiểm tra được lẫn nhau, nếu chỉ một bit chẵn lẻ bị sai thôi, trong khi tất cả các bit khác là đúng, thì chỉ có bit chẵn

lẻ nói đến là sai mà thôi và không phải là các bit nó kiểm tra (not any bit it checks).

Cuối cùng, giả sử có hai bit biến đổi, tại vị trí x và y. Nếu x và y có cùng một bit tại vị trí 2^k trong đại diện nhị phân của chúng, thì bit chẵn lẻ tương ứng với vị trí đấy kiểm tra cả hai bit, và do đó sẽ giữ nguyên giá trị, không thay đổi. Song một số bit chẵn lẻ nào đấy nhất định phải bị thay đổi, vì $x \neq y$, và do đó hai bit tương ứng nào đó có giá trị x và y khác nhau. Do vậy, mã Hamming phát hiện tất cả các lỗi do hai bit bị thay đổi — song nó không phân biệt được chúng với các lỗi do 1 bit bị thay đổi.

Chương 7

Kết luận

Bài báo cáo đã hoàn thành trình bày kiến thức cơ sở và những tính chất quan trọng của ngôn ngữ chính quy. Giải quyết được khá nhiều bài toán liên quan đến việc biểu diễn một ngôn ngữ chính quy. Tuy nhiên vẫn còn nhiều vấn đề chưa được giải quyết như việc rút gọn biểu thức thức chính quy, hay một số thuật toán được miêu tả trong chương 4 vẫn có độ phức tạp mũ. Mặc dù thuật toán Sardinas-Partison được trình bày trong phần kiểm định mã có thể mở rộng cho trường hợp tập X là tập vô hạn có thể nhận biết được (recognizable set), chương 5 đã trình bày sơ lược về lý thuyết mã khá hoàn chỉnh. Bên cạnh các vấn đề về lý thuyết, bài báo cáo cũng trình bày về mã sửa sai - một ứng dụng khá quan trọng lý thuyết mã hóa. Lời cuối cùng, nhóm chúng em xin cảm ơn cô Ngô Thị Hiền đã giúp đỡ và hướng dẫn nhóm chúng em hoàn thiện bài báo này.

Tài liệu tham khảo

[1] Introduction to Automata Theory, Languages, and Computation, *John E. Hopcroft, Rajeev Motwani, Jeffray D.Ullman*

[2] A Necessary and Sufficient Pumping Lemma for Regular Languages, *Jeffrey Jaffe, 1978*

Phân công công việc

Nguyễn Anh Tú

1. Tìm hiểu lý thuyết, và thực hiện báo cáo chương 2,3,5
2. Làm giao diện chương trình

Phạm Anh Tuấn

1. Lập trình 9 thuật toán, và viết báo cáo chương 4
2. Đọc và tìm hiểu và mã sửa sai (chương 6)