CHƯƠNG V
Các ngôn ngữ phi ngữ cảnh

Nội dung của chương năm đề cập đến:

- ✓ Cây dẫn xuất và các phương pháp xác định văn phạm phi ngữ cảnh,
- ✓ Tính nhập nhằng và quá trình giản lược văn phạm phi ngữ cảnh,
- ✓ Các dạng chuẩn: dạng chuẩn Chomsky và dạng chuẩn Greibach,
- ✓ Bổ đề Bơm (điều kiện cần) cho các ngôn ngữ phi ngữ cảnh và một số thuật toán quyết định được.

5.1 Các ngôn ngữ phi ngữ cảnh và các cây dẫn xuất

Ngôn ngữ phi ngữ cảnh thường được sử dụng trong thiết kế các bộ chương trình phân tích cú pháp. Nó cũng được sử dụng để mô tả các khối cấu trúc trong các ngôn ngữ lập trình. Trong phần này chúng ta sử dụng cây dẫn xuất để biểu diễn cho ngôn ngữ phi ngữ cảnh.

Trước tiên chúng ta nhắc lại định nghĩa của văn phạm phi ngữ cảnh. G là phi ngữ cảnh nếu mọi qui tắc dẫn xuất đều có dạng:

$$A \to \alpha, \text{ trong đó } A \in V_N \text{ và } \alpha \in (V_N \cup \Sigma)^*$$

Ví dụ 5.1 Xây dựng một văn phạm phi ngữ cảnh G để sinh ra tất cả các số nguyên.

Giải. Giả sử G = (V_N, Σ, P, S), trong đó

V_N = {S, <dau>, <chu so>, <So nguyen>}

Σ = {0, 1, 2, . . ., 9, +, - }

P bao gồm các qui tắc: S → <dau><So nguyen>,

<dau> → + | -,

<So nguyen> → <chu so> <So nguyen> | <chu so>

<chu so> → 0 | 1 | 2 | . . .| 9.

Dễ dàng kiểm chứng được là L(G) là tập tất cả các số nguyên.

Mỗi dẫn xuất của một văn phạm có thể biểu diễn dưới dạng một cây, dc gọi là cây dẫn xuất.

Định nghĩa 5.1 Cây dẫn xuất (còn gọi là *cây phân tích cú pháp*) cho văn phạm phi ngữ cảnh G = (V_N, Σ, P, S) là cây thoả mãn các tính chất sau:

(i) Các đỉnh đều được gán nhãn là biến, ký tự kết thúc hoặc Λ,

(ii) Gốc có nhãn S,

(iii) Nhãn các đỉnh bên trong (không phải là lá) là các biến,

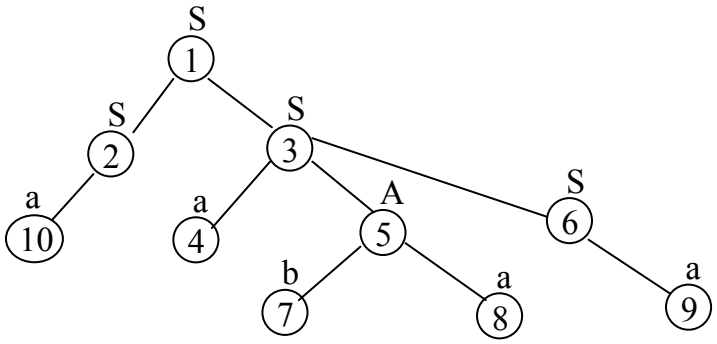
(iv) Nếu các đỉnh $n_1, n_2, ..., n_k$ được viết với các nhãn $X_1, X_2, ..., X_k$ là các con của đỉnh n có nhãn A thì trong P có tương ứng qui tắc $A \rightarrow X_1X_2...X_k$

(v) Đỉnh *n* là lá nếu nhãn của nó là $a \in \Sigma$ hoặc là ký hiệu rỗng Λ.

Ví dụ 5.2 Cho trước văn phạm G = ({S, A}, {a, b}, P, S), trong đó P gồm:

$S \rightarrow aAS \mid a \mid SS, A \rightarrow SbA \mid ba$

Cây dẫn xuất của G để sinh ra xâu w = *aabaa* sẽ là:



Hình H5-1 Cây dẫn xuất cho văn phạm G để sinh ra xâu w = aabaa

Sắp xếp các đỉnh lá từ trái qua phải

+ Các đỉnh con của đỉnh gốc được sắp xếp từ trái qua phải, nghĩa là các đỉnh ở mức 1 được sắp xếp từ bên trái,

+ Nếu hai đỉnh v_1, v_2 ở mức 1 và v_1 ở bên trái của v_2 thì ta nói rằng v_1 bên trái của tất cả các đỉnh con của v_2 và các đỉnh con của v_1 cũng là ở bên trái của v_2, đồng thời cũng ở bên trái của các đỉnh con của v_2.

+ Lặp lại quá trình trên cho mức 2, 3, ..., k.

Điều mà chúng ta quan tâm nhất là sự sắp xếp các lá của cây dẫn xuất.

Ví dụ: trong cây ở Hình H5-1 thì các đỉnh lá được sắp xếp từ trái 10-4-7-8-9 cho giá trị là xâu *aabaa* và xâu này được sinh ra bởi G.

Định nghĩa 5.2 *Kết quả* của một cây dẫn xuất là dãy ghép các nhãn của các đỉnh lá từ trái qua phải.

Ví dụ: *Kết quả* của cây dẫn xuất ở Hình H5-1 là *aabaa*.

Sử dụng các qui tắc của văn phạm G:

$S \Rightarrow SS \Rightarrow aS \Rightarrow aaAS \Rightarrow aabaS \Rightarrow aabaa$

Vậy kết quả của một cây dẫn xuất cũng là một từ được sinh ra bởi văn phạm G.

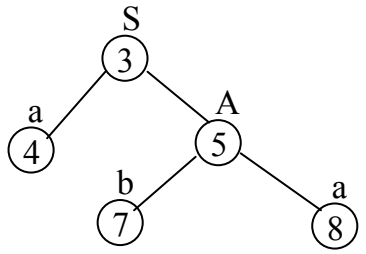
Định nghĩa 5.3 *Cây con của cây dẫn xuất T* là một cây có

(i) Gốc của nó là một đỉnh *v* của cây T,

(ii) Các đỉnh của nó cũng là con cháu của *v* cùng với các nhãn tương ứng,

(iii) Các cung nối các con cháu tương ứng của *v*.

Ví dụ: cây trong hình H5-2 là cây con của cây ở hình H5-1



Hình H5-2 Một cây con của cây T ở hình H5-1

Lưu ý: Cây con cũng giống như cây dẫn xuất, chỉ khác nhau là nhãn của cây con có thể không phải là ký hiệu bắt đầu S. Nếu nhãn của gốc của cây con là A thì cây con đó được gọi là A-cây.

Định lý 5.1 Giả sử $G = (V_N, \Sigma, P, S)$ là văn phạm phi ngữ cảnh. Khi đó

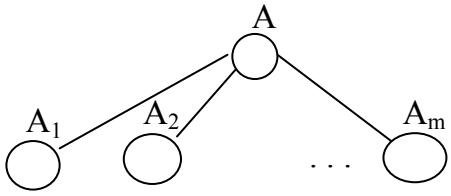
$S \overset{*}{\Rightarrow} \alpha$ khi và chỉ khi tồn tại cây dẫn xuất T của G để T có kết quả là α.

Chứng minh:

Chỉ cần chứng minh rằng $A \overset{*}{\Rightarrow} \alpha$ khi và chỉ khi A-cây có kết quả là α. (4.17)

Giả sử α là kết quả A-cây T_1. Chúng ta chứng minh $A \overset{*}{\Rightarrow} \alpha$ bằng phương pháp qui nạp theo số các đỉnh bên trong của T_1.

Cơ sở qui nạp: Nếu T_1 chỉ có một đỉnh bên trong, nghĩa là tất cả các đỉnh con của gốc đều là lá. Khi đó cây T_1 có dạng:



Hình H5-3 (a) A-Cây có một đỉnh bên trong

Theo (iv) trong định nghĩa 5.1 suy ra $A \to A_1A_2...A_m = \alpha$ là một qui tắc trong G. Vậy $A \Rightarrow \alpha$.

Giả thiết qui nạp: Giả sử (4.17) đúng với các cây con có k – 1 đỉnh bên trong, k > 1.

Chúng ta xét A-cây T_1 có k đỉnh bên trong ($k \geq 2$). Giả thiết gốc của T_1 có $v_1, v_2, ..., v_m$ là các đỉnh con được gắn nhãn tương ứng $X_1, X_2, ..., X_m$. Theo (iv) trong định nghĩa 5.1 suy ra $A \to X_1X_2...X_m$ là một qui tắc trong P. Do vậy,

$$A \Rightarrow X_1X_2...X_m \qquad (4.18)$$

Bởi vì $k \geq 2$ nên trong số các đỉnh con của A có ít nhất một đỉnh là đỉnh bên trong. Nếu v_i là một đỉnh bên trong thì lại xét tiếp cây con của T_1 có gốc chính là v_1. Hiển

nhiên là số các đỉnh bên trong của cây con gốc v_i là nhỏ hơn k, vậy theo giả thiết qui nạp suy ra $X_i \overset{*}{\Rightarrow} \alpha_i$, X_i là nhãn của v_i.

Nếu v_i không phải là đỉnh bên trong, thì $X_i = \alpha_i$.

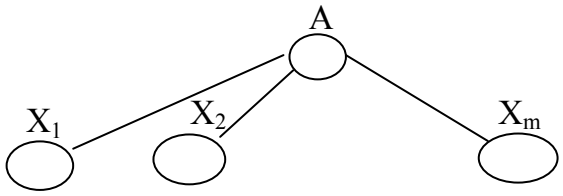
Sử dụng (5.1) chúng ta nhận được:

$$A \Rightarrow X_1X_2...X_m \overset{*}{\Rightarrow} \alpha_1X_2...X_m \ldots \overset{*}{\Rightarrow} \alpha_1\alpha_2 \ldots \alpha_m = \alpha$$

Nghĩa là $A \overset{*}{\Rightarrow} \alpha$.

Để chứng minh chiều ngược lại, chúng ta giả thiết rằng $A \overset{*}{\Rightarrow} \alpha$. Chúng ta đi xây dựng A-cây để có kết quả là α. Thực hiện điều này bằng phương pháp qui nạp theo số bước trong dẫn xuất $A \overset{*}{\Rightarrow} \alpha$.

Khi $A \Rightarrow \alpha$, $A \rightarrow \alpha$ là qui tắc trong P. Nếu $\alpha = X_1X_2...X_m$, A-cây cho kết quả α có thể xây dựng dễ dàng và có dạng như hình H5-3 (b).



Hình H5-3 (b) Cây dẫn xuất cho dẫn xuất một bước

Giả sử dẫn xuất $A \overset{*}{\Rightarrow} \alpha$ thực hiện qua k bước, k > 1. Khi đó ta có thể chia nó thành

$A \Rightarrow X_1X_2...X_m \overset{k-1}{\Rightarrow} \alpha$. Dẫn xuất $A \Rightarrow X_1X_2...X_m$ kéo theo $A \rightarrow X_1X_2...X_m$ là một qui tắc trong P. Còn trong dẫn xuất $X_1X_2...X_m \overset{k-1}{\Rightarrow} \alpha$ thì:

(i) Hoặc X_i không chuyển trạng thái suốt trong quá trình dẫn xuất, $X_i = \alpha_i$

(ii) Hoặc X_i biến đổi trong quá trình dẫn xuất. Giả sử α_i là xâu nhận được từ X_i, $X_i \overset{*}{\Rightarrow} \alpha_i$.

Cây dẫn xuất cho kết quả $\alpha = \alpha_1\alpha_2 \ldots \alpha_m$ được xây dựng như sau:

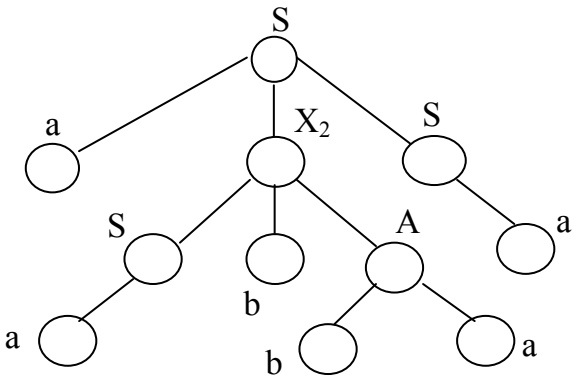
Khi $A \rightarrow X_1X_2...X_m$ là một qui tắc trong P, ta xây dựng cây có m lá: X_1, X_2, ..., X_m. Từ X_i suy dẫn ra α_i được thực hiện nhỏ hơn k bước, vậy theo giả thiết qui nạp thì X_i-cây sẽ cho kết quả là α_i. Do đó, A-cây sẽ cho kết quả là α. ■

Ví dụ 5.3 Xét văn phạm có các qui tắc $S \rightarrow aAS \mid a$, $A \rightarrow SbA \mid SS \mid ba$. Chỉ ra rằng $S \overset{*}{\Rightarrow}$ aabbaa và xây dựng cây dẫn xuất để cho kết quả *aabbaa*.

Giải:

$S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS \Rightarrow a^2bbaS \Rightarrow a^2b^2a^2$, nghĩa $S \overset{*}{\Rightarrow} a^2b^2a^2$.

Cây dẫn xuất cho kết quả trên là:



Hình H5-4 Cây dẫn xuất cho kết quả $a^2b^2a^2$

Trong các dẫn xuất ở trên chúng ta nhận thấy biến bên trái nhất luôn có thể áp dụng một qui tắc nào đó trong P để dẫn xuất tiếp.

Định nghĩa 5.3 Dẫn xuất $A \overset{*}{\Rightarrow} w$ là *dẫn xuất trái nhất* nếu chỉ áp dụng qui tắc dẫn xuất cho biến bên trái nhất trong mọi bước dẫn xuất.

Định nghĩa 5.4 Dẫn xuất $A \overset{*}{\Rightarrow} w$ là *dẫn xuất phải nhất* nếu chỉ áp dụng qui tắc dẫn xuất cho biến bên phải nhất trong mọi bước dẫn xuất.

Lưu ý: Trong ví dụ 5.3, xét dẫn xuất $S \Rightarrow aAS$, biến bên trái nhất của dẫn xuất này là A. Mặt khác trong G trên có qui tắc $A \rightarrow SbA$, áp dụng vào dẫn xuất ta có dẫn xuất trái nhất $aAS \Rightarrow aSbAS$.

Định lý 5.2 Nếu $A \overset{*}{\Rightarrow} w$ là một dẫn xuất trong văn phạm phi ngữ cảnh G thì sẽ tồn tại một dẫn xuất trái nhất cho w.

Chứng minh:

Chúng ta chứng minh bằng qui nạp theo số bước dẫn xuất trong $A \overset{*}{\Rightarrow} w$.

Bước cơ sở: Nếu chỉ dẫn xuất qua một bước thì $A \Rightarrow w$, nghĩa là vế trái chỉ có một biến, suy ra đó là dẫn xuất trái nhất.

Giả thiết: Định lý 5.2 đúng với *k* bước dẫn xuất, nghĩa là nếu $A \overset{k}{\Rightarrow} w$ thì tồn tại dẫn xuất bên trái nhất để dẫn ra w.

Xét $A \overset{k+1}{\Rightarrow} w$. Ta có thể chia dẫn xuất này thành $A \Rightarrow X_1X_2...X_m \overset{k}{\Rightarrow} w$. Tương tự, có thể tách $w = w_1w_2 \ldots w_m$ và $X_i \overset{*}{\Rightarrow} w_i$ qua nhiều nhất là k bước. Theo giả thiết qui nạp, tất cả các dẫn xuất $X_i \overset{*}{\Rightarrow} w_i$ đều tồn tại dẫn xuất bên trái nhất, do vậy:

$$A \Rightarrow X_1X_2...X_m \overset{*}{\Rightarrow} w_1X_2...X_m \overset{*}{\Rightarrow} w_1w_2X_3...X_m \overset{*}{\Rightarrow} w_1w_2...w_m = w \qquad \blacksquare$$

Hệ quả 5.1 Mọi cây dẫn xuất của w đều tạo ra dẫn xuất trái nhất của w.

Ví dụ 5.4 Cho trước văn phạm G có các qui tắc S $\rightarrow$ 0B | 1A, A $\rightarrow$ 0 | 0S | 1AA, B $\rightarrow$ 1 | 1S | 0BB và cho biết w = 00110101.

(i) Tìm dẫn xuất trái nhất,

(ii) Tìm dẫn xuất phải nhất,

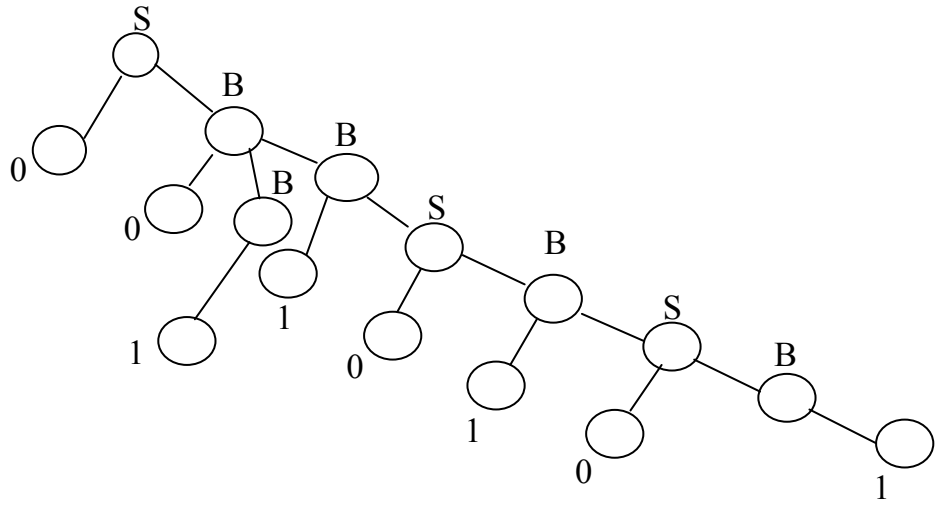
(iii) Xác định cây dẫn xuất.

Giải:

(i) $S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 0011S \Rightarrow 0^2 1^2 0B \Rightarrow 0^2 1^2 01S \Rightarrow 0^2 1^2 010B \Rightarrow 0^2 1^2 0101$

(ii) $S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 00B1S \Rightarrow 0^2 B10B \Rightarrow 0^2 B101S \Rightarrow 0^2 B1010B \Rightarrow 0^2 1B10101 \Rightarrow 0^2 1^2 0101$

(iii) Cây dẫn xuất cho $0^2 1^2 0101$ là



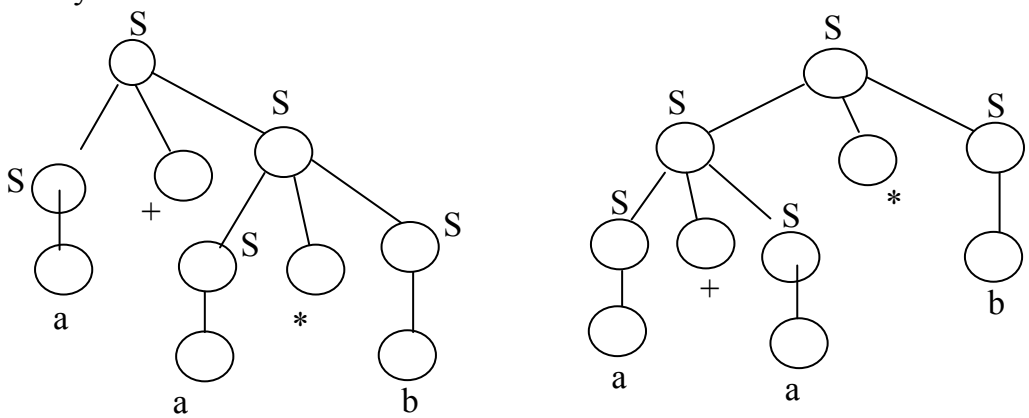
Hình H5-5 Cây dẫn xuất cho kết quả $0^2 1^2 0101$

5.2 Sự nhập nhằng trong văn phạm phi ngữ cảnh

Đôi khi chúng ta gặp phải sự nhập nhằng trong dẫn xuất các xâu trong một văn phạm G.

Định nghĩa 5.5 w $\in$ L(G) được gọi là nhập nhằng nếu tồn tại nhiều hơn một cây dẫn xuất sinh ra w (hoặc có nhiều hơn một dẫn xuất trái nhất).

Ví dụ 5.4 Xét G = ({S}, {a, b, +, *}, P, S), trong đó P có S $\rightarrow$ S + S | S * S | a | b. Có hai cây dẫn xuất cho a + a * b:



Hình H5-6 Hai cây dẫn xuất cùng cho kết quả a + a * b

Định nghĩa 5.6 Văn phạm phi ngữ cảnh G là nhập nhằng nếu có $w \in L(G)$ là nhập nhằng.

Ví dụ: Văn phạm ở ví dụ 5.4 là nhập nhằng.

5.3 Giản lược các văn phạm phi ngữ cảnh

Trong văn phạm phi ngữ cảnh có thể có nhiều yếu tố dư thừa, ví dụ có những ký hiệu trong $V_N \cup \Sigma$ không được sử dụng, không xuất hiện trong các qui tắc của P, hoặc có những qui tắc dạng $A \to B$ chỉ làm mất thời gian suy diễn mà không cho được những thông tin gì mới.

Ví dụ 5.5 Xét văn phạm G = ({S, A, B, C, E }, {a, b, c}, P, S), trong đó

$P = \{S \to AB, A \to a, B \to b, B \to C, E \to c \mid \Lambda\}$

Dễ nhận thấy là L(G) = {ab} và G có các dư thừa sau cần phải loại bỏ:

(i) Những biến không dẫn ra các xâu kết thúc, ví dụ C.

(ii) Những biến không xuất hiện ở vế phải, hoặc không xuất hiện ở vế trái của các qui tắc, ví dụ E không xuất hiện ở vế phải và không dẫn đến được từ ký hiệu bắt đầu S.

(iii) Các qui tắc rỗng, ví dụ $E \to \Lambda$.

(iv) Các qui tắc dạng $A \to B$ (suy diễn giữa hai biến với nhau), ví dụ $B \to C$.

Văn phạm rút gọn sẽ là G′ = ({S, A, B}, {a, b}, P′, S) với

$P' = \{S \to AB, A \to a, B \to b\}$

Hiển nhiên, L(G′) = L(G).

Lưu ý: Những biến sau được gọi là các biến dư thừa

+ Những biến không dẫn ra được xâu kết thúc được gọi là biến vô sinh.

+ Những biến không dẫn xuất ra được từ S được gọi là không dẫn đến được.

Vì lẽ đó cần loại bỏ các biến dư thừa mà không làm ảnh hưởng tới quá trình sinh ra các ngôn ngữ của văn phạm phi ngữ cảnh.

5.3.1 Lược giản các văn phạm phi ngữ cảnh

Định lý 5.3 Nếu G là văn phạm phi ngữ cảnh và $L(G) \neq \varnothing$ thì có thể tìm được văn phạm G′ tương đương sao cho mọi biến trong G′ đều có thể dẫn xuất ra xâu kết thúc.

Chứng minh: Cho văn phạm $G = (V_N, \Sigma, P, S)$.

Chúng ta định nghĩa $G' = (V_N', \Sigma, P', S)$ như sau:

(i) *Xây dựng V_N'.* Định nghĩa $W_i \subseteq V_N$ bằng đệ qui:

+ $W_1 = \{ A \in V_N \mid$ nếu tồn tại $A \to w$ và $w \in \Sigma^*\}$

$$+ W_{i+1} = W_i \cup \{ A \in V_N \mid \text{nếu tồn tại } A \to \alpha \text{ và } \alpha \in (\Sigma \cup W_i)^* \}$$

Theo định nghĩa đệ qui trên thì $W_i \subseteq W_{i+1}$ với mọi i. Bởi vì V_N là hữu hạn, nên tồn tại $k \le |V_N|$ để $W_i = W_{i+1}$. Cuối cùng đặt $V_N' = W_i$.

(ii) Kiến thiết P′. $P' = \{A \to \alpha \mid \Lambda, \alpha \in (\Sigma \cup V_N')^*\} \cap P$

Chúng ta cần chỉ ra G′ chính là văn phạm cần tìm.

Trước tiên, S thuộc V_N', vì nếu $S \notin V_N'$ thì $L(G) = \varnothing$, điều này mâu thuẫn với giả thiết là $L(G) \ne \varnothing$.

Tiếp theo cần chứng minh:

(i) Nếu $A \in V_N'$ thì $A \overset{*}{\underset{G'}{\Rightarrow}} w$, $w \in \Sigma^*$ và ngược lại nếu $A \overset{*}{\underset{G}{\Rightarrow}} w$ thì $A \in V_N'$

(ii) $L(G') = L(G)$.

Để chứng minh (i) chúng ta lưu ý rằng $W_k = W_1 \cup W_2 \ldots \cup W_k$. Chứng minh qui nạp theo i = 1, 2, ..., k, $A \in W_i$ kéo theo $A \overset{*}{\underset{G'}{\Rightarrow}} w$ với $w \in \Sigma^*$.

Nếu $A \in W_1$ thì $A \to w$ và $w \in \Sigma^*$, do đó $A \underset{G}{\Rightarrow} w$. Theo định nghĩa thì $A \to w$ cũng thuộc P′ (cơ sở qui nạp đúng).

Giả sử nó đúng với i < k.

Xét $A \in W_{i+1}$. Theo định nghĩa thì

+ Hoặc $A \in W_i$. Khi đó $A \overset{*}{\underset{G'}{\Rightarrow}} w$, $w \in \Sigma^*$ theo giả thiết qui nạp.

+ Hoặc tồn tại $A \to \alpha$ và $\alpha \in (\Sigma \cup W_i)^*$. Theo định nghĩa thì $A \to \alpha$ sẽ thuộc P′. Ta có thể viết $\alpha = X_1X_2 \ldots X_m$, $X_j \in \Sigma \cup W_i$. Nếu $X_j \in W_i$ thì theo giả thiết qui nạp suy ra $X_j \overset{*}{\underset{G'}{\Rightarrow}} w_j$, $w_j \in \Sigma^*$. Cuối cùng dẫn tới

$A \overset{*}{\underset{G'}{\Rightarrow}} w_1w_2 \ldots w_m \in \Sigma^*$.

Chiều ngược lại chứng minh tương tự qui nạp theo các bước suy dẫn trong dẫn xuất $A \underset{G}{\Rightarrow} w$.

Để chứng minh (ii) thì chúng ta thấy ngay $L(G') \subseteq L(G)$ bởi vì $V_N' \subseteq V_N$ và $P' \subseteq P$.

Chứng minh chiều ngược lại phải dựa vào bổ đề:

$$A \overset{*}{\underset{G'}{\Rightarrow}} w \text{ nếu } A \overset{*}{\underset{G}{\Rightarrow}} w \text{ và } w \in \Sigma^* \qquad (4.19)$$

Bổ đề này có thể chứng qui nạp theo số bước trong dẫn xuất $A \overset{*}{\underset{G}{\Rightarrow}} w$.

Áp dụng (4.19) cho trường hợp A = S suy ra $L(G) \subseteq L(G')$. ■

Định lý 5.4 (Loại bỏ các ký hiệu không dẫn đến được)

Cho trước văn phạm phi ngữ cảnh $G = (V_N, \Sigma, P, S)$, chúng ta xây dựng văn phạm $G' = (V_N', \Sigma', P', S)$ tương đương sao cho với mọi ký hiệu $X \in V_N' \cup \Sigma'$ đều tồn tại $\alpha, \beta \in V_N' \cup \Sigma'$ để $S \overset{*}{\Rightarrow} \alpha X \beta$, nghĩa là mọi biến đều dẫn đến xâu kết thúc và xuất phát từ S.

Chứng minh: $G' = (V_N', \Sigma', P', S)$ được xây dựng như sau

(a) Xây dựng W_i với $i \geq 1$

+ $W_1 = \{S\}$

+ $W_{i+1} = W_i \cup \{X \in V_N \cup \Sigma \mid$ nếu tồn tại $A \to \alpha$ và $\alpha \in W_i$ và α chứa $X\}$

Tương tự như trên $W_i \subseteq V_N \cup \Sigma$ và $W_i \subseteq W_{i+1}$ nên sẽ tồn tại k để $W_k = W_{k+1}$.

+ V_N', Σ', P' được xây dựng như sau:

$$V_N' = V_N \cap W_k,\ \Sigma' = \Sigma \cap W_k \text{ và } P' = \{A \to \alpha \mid A \to \alpha \in P,\ A \in W_k\}$$

Với cách xây dựng chi tiết tương tự như trên, chúng ta có $L(G') = L(G)$. ■

Định nghĩa 5.7 Văn phạm phi ngữ cảnh $G = (V_N, \Sigma, P, S)$ được gọi là *tối giản* (không có ký hiệu dư thừa) nếu mọi ký hiệu trong $V_N \cup \Sigma$ đều xuất hiện ít nhất trong một dẫn xuất dẫn đến xâu kết thúc. Nghĩa là, với mọi X trong $V_N \cup \Sigma$, tồn tại một dẫn xuất $S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w \in L(G)$.

Định lý 5.5 Với mọi văn phạm phi ngữ cảnh G luôn tồn tại văn phạm G′ tối giản tương đương.

Chứng minh: Xây dựng văn phạm không dư thừa theo hai bước.

Bước 1. Xây dựng văn phạm G_1 tương đương với G sao cho mọi biến đều dẫn xuất ra các xâu kết thúc (định lý 5.3).

Bước 2. Xây dựng $G' = (V_N', \Sigma', P', S)$ tương đương với G_1 sao cho mọi ký hiệu trong $V_N' \cup \Sigma'$ đều dẫn đến được (định lý (5.4).

Ví dụ 5.6 Tìm văn phạm không dư thừa tương đương với văn phạm G có các qui tắc sau:

$$S \to AB \mid CA,\ B \to BC \mid AB,\ A \to a,\ C \to aB \mid b$$

Giải:

Bước 1. $W_1 = \{A, C\}$ bởi $A \to a$ và $C \to b$

$W_2 = \{A, C\} \cup \{A_1 \mid A_1 \to \alpha, \alpha \in (\Sigma \cup \{A, C\}^*\}$

$= \{A, C\} \cup \{S\}$ bởi $S \to CA$

$W_3 = \{A, C, S\} \cup \{A_1 \mid A_1 \to \alpha, \alpha \in (\Sigma \cup \{A, C, S\}^*\}$

$= \{A, C, S\} \cup \varnothing$

Vậy $W_3 = W_2$. Do đó,

$V_N' = W_2 = \{S, A, C\}$

$P' = \{A_1 \to \alpha, \alpha \in (V_N' \cup \{a, b\}^*\} \cap P$

$= \{S \to CA, A \to a, C \to b\}$

Kết thúc bước 1: $G_1 = \{\{S, A, C\}, \{a, b\}, \{S \to CA, A \to a, C \to b\}, S\}$

Bước 2. Áp dụng định lý 5.4 cho G_1.

$W_1 = \{S\}$

$W_2 = \{S\} \cup \{A, C\}$ bởi vì $S \to CA$ và $S \in W_1$.

$W_3 = \{S, A, C\} \cup \{a, b\}$ bởi vì $A \to a$, $C \to b$ và $A, C \in W_2$.

Tương tự $W_4 = W_3 = V_N' \cup \Sigma$, đồng thời $P'' = \{A_1 \mid A_1 \to \alpha, A_1 \in W_3\} = P'$.

Vậy, $G' = \{\{S, A, C\}, \{a, b\}, \{S \to CA, A \to a, C \to b\}, S\}$ là không dư thừa.

5.3.2 Loại bỏ các qui tắc rỗng

Văn phạm phi ngữ cảnh có thể có các qui tắc dạng $A \to \Lambda$ được gọi là *qui tắc rỗng*. Qui tắc này chỉ sử dụng để xoá các biến trong quá trình dẫn xuất. Trong phần này chúng ta muốn loại bỏ những qui tắc rỗng.

Định nghĩa 5.8 Biến A trong văn phạm phi ngữ cảnh được gọi là *biến được rỗng hoá* nếu $A \overset{*}{\Rightarrow} \Lambda$.

Định lý 5.6 Nếu $G = (V_N, \Sigma, P, S)$ là văn phạm phi ngữ cảnh thì có thể tìm được văn phạm phi ngữ cảnh G_1 không có qui tắc rỗng sao cho $L(G_1) = L(G) - \Lambda$.

Chứng minh: Xây dựng $G_1 = (V_N, \Sigma, P', S)$ như sau:

Bước 1. Xác định tập W các biến rỗng hóa

+ $W_1 = \{A \in V_N \mid A \to \Lambda \text{ trong } P\}$

+ $W_{i+1} = W_i \cup \{ A \in V_N \mid \text{nếu tồn tại } A \to \alpha \text{ và } \alpha \in W_i^* \}$

Vì V_N hữu hạn và $W_i \subseteq W_{i+1}$ nên tồn tại $k \le |V_N|$ để $W_i = W_{i+1}$. Đặt $W = W_i$ là tập tất cả các biến có thể rỗng hoá.

Bước 2. Kiến thiết P′:

+ Mọi qui tắc mà vế phải không có biến rỗng hoá sẽ được đưa vào P′ .

+ Nếu $A \to X_1X_2...X_k$ thuộc P thì bổ sung các qui tắc dạng: $A \to \alpha_1\alpha_2...\alpha_k$, trong đó $\alpha_i = X_i$ nếu $X_i \notin W$, hoặc $\alpha_i = X_i, \Lambda$ nếu $X_i \in W$ và $\alpha_1\alpha_2...\alpha_k \ne \Lambda$.

Đến đây thì $G_1 = (V_N, \Sigma, P', S)$ sẽ không còn chứa các qui tắc rỗng và có thể chứng minh được rằng $L(G_1) = L(G) - \Lambda$. ■

Hệ quả 5.2 Tồn tại thuật toán kiểm tra xem $\Lambda \in L(G)$ với G là văn phạm phi ngữ cảnh.

Chứng minh: $\Lambda \in L(G) \Leftrightarrow S \in W$, nghĩa là S là được rỗng hoá, W được xây dựng như trong phần chứng minh định lý 5.6. Theo định lý trên, việc xây dựng tập W là kết thúc sau hữu hạn bước (nhiều nhất là $|V_N|$ bước). Vậy thuật toán kiểm tra $\Lambda \in L(G)$ được thực hiện như sau:

(i) Xây dựng W

(ii) Kiểm tra xem $S \in W$.

Hệ quả 5.3 Nếu $G = (V_N, \Sigma, P, S)$ là văn phạm phi ngữ cảnh thì luôn tìm được văn phạm phi ngữ cảnh $G_1 = (V_N', \Sigma, P_1, S_1)$ không có qui tắc rỗng, ngoại trừ qui tắc $S_1 \to \Lambda$ khi $\Lambda \in L(G)$. Khi $S_1 \to \Lambda$ thuộc P_1 thì S_1 không xuất hiện ở vế phải của các qui tắc.

Chứng minh: Từ hệ quả 5.2 chúng ta biết được Λ thuộc L(G) hay không.

a/ Nếu Λ không thuộc L(G) thì G_1 nhận được từ định lý 5.6 là tương đương với G.

b/ Nếu $\Lambda \in L(G)$ thì xây dựng $G' = (V_N, \Sigma, P', S)$ theo định lý 5.6 và chúng ta có $L(G') = L(G) - \Lambda$. Sau đó định nghĩa $G_1 = (V_N \cup \{S_1\}, \Sigma, P_1, S_1)$, trong đó $P_1 = P' \cup \{S_1 \to S, S_1 \to \Lambda\}$. Hiển nhiên là S_1 không nằm trong V_N và do vậy sẽ không xuất hiện ở vế phải của mọi qui tắc của P_1, đồng thời $L(G_1) = L(G)$.

5.3.3 Loại bỏ các qui tắc đơn vị

Trong văn phạm phi ngữ cảnh có thể có các qui tắc dạng $A \to B$, $A, B \in V_N$ và cần phải loại bỏ chúng.

Định nghĩa 5.9 *Qui tắc đơn vị* (qui tắc dây chuyền) trong văn phạm phi ngữ cảnh là qui tắc dạng $A \to B$ với A, B là các biến $\in V_N$.

Định lý 5.7 Nếu $G = (V_N, \Sigma, P, S)$ là văn phạm phi ngữ cảnh thì luôn tìm được văn phạm phi ngữ cảnh G_1 không có qui tắc rỗng và không có các qui tắc đơn vị sao cho $L(G_1) = L(G)$.

Chứng minh: Áp dụng hệ quả 5.3 chúng ta xây dựng được $G' = (V_N', \Sigma, P_1, S_1)$ không có qui tắc rỗng và $L(G') = L(G)$.

Để khử các qui tắc đơn vị suy được từ biến A, chúng ta làm như sau:

Bước 1: Xác định tập các biến được dẫn xuất từ A. Định nghĩa $W_i(A)$ đệ qui như sau:

$$W_0(A) = \{A\}$$

$$W_{i+1}(A) = W_i(A) \cup \{B \in V_N \mid C \to B \text{ thuộc } P \text{ với } C \in W_i(A)\}$$

Tương tự như trên, quá trình này dừng sau $k \le |V_N|$ bước, $W_{k+1}(A) = W_k(A)$. Đặt $W(A) = W_k(A)$ là tập tất cả các biến được dẫn xuất từ A.

Bước 2: Kiến thiết các A-dẫn xuất trong G_1. Các A-dẫn xuất trong G_1 có thể là:

+ Những qui tắc không phải dạng đơn vị trong G'

+ $A \to \alpha$ khi $B \to \alpha$ là qui tắc của G với $B \in W(A)$ và $\alpha \notin V_N$.

Văn phạm cần xây dựng là $G_1 = (V_N, \Sigma, P_1, S)$, trong đó P_1 được xây dựng như bước 2. cho mọi biến $A \in V_N$.

Sử dụng phương pháp qui nạp chúng ta chứng minh được rằng $L(G_1) = L(G)$. ■

Ví dụ 5.7 Cho trước văn phạm G có các qui tắc $S \to AB$, $A \to a$, $B \to C \mid b$, $C \to D$, $D \to E$, $E \to a$. Tìm văn phạm tương đương không có các qui tắc đơn vị.

Giải:

Bước 1. $W_0(S) = \{S\}$, $W_1(S) = W_0(S) \cup \varnothing$. Vậy $W(S) = \{S\}$.

$W(A) = \{A\}$, $W(E) = \{E\}$,

$W_0(B) = \{B\}$, $W_1(B) = W_0(B) \cup \{C\} = \{B, C\}$,

$W_2(B) = W_1(B) \cup \{D\} = \{B, C, D\}$,

$W_3(B) = W_2(B) \cup \{E\} = \{B, C, D, E\}$ và $W_4(B) = W_3(B)$.

Vậy $W(B) = \{B, C, D, E\}$. Tương tự:

$W(C) = \{C, D, E\}$, $W(D) = \{D, E\}$.

Bước 2: Xây dựng G_1 có các qui tắc được xác định như bước 2. của định lý 5.7:

$S \to AB$, $A \to a$, $B \to b \mid a$, $E \to a$, $C \to a$, $D \to a$

Hiển nhiên G_1 không chứa qui tắc đơn vị và $L(G_1) = L(G)$, nhưng vẫn còn chứa những biến không dẫn đến được có thể loại bỏ đi như C, D, E.

5.4 Các dạng chuẩn của văn phạm phi ngữ cảnh

Trong định nghĩa văn phạm phi ngữ cảnh, vế phải của một qui tắc có thể là một xâu bất kỳ có cả các biến và các ký hiệu kết thúc. Khi các qui tắc thoả mãn một số ràng buộc chúng ta có thể đưa các văn phạm đó về dạng chuẩn.

5.4.1 Dạng chuẩn Chomsky

Định nghĩa 5.8 Văn phạm phi ngữ cảnh G ở *dạng chuẩn Chomsky* (viết tắt là CNF) nếu các qui tắc đều có dạng:

$A \to a$ hoặc $A \to BC$, và $S \to \Lambda$ nếu $\Lambda \in L(G)$, với $S, A, B, C \in V_N$, $a \in \Sigma$.

Khi Λ thuộc L(G) thì S không xuất hiện ở vế phải của mọi qui tắc.

Ví dụ: Văn phạm G có các qui tắc $S \to AB \mid \Lambda$, $A \to a$, $B \to b$ là ở dạng CNF.

Nhận xét: Ở dạng CNF, các vế phải của mọi qui tắc chỉ có thể là một ký hiệu kết thúc (hoặc ký hiệu rỗng Λ) hoặc hai biến.

Vấn đề quan trọng đặt ra là có thể chuyển một văn phạm phi ngữ cảnh bất kỳ về CNF? Định lý sau trả lời cho câu hỏi này.

Định lý 5.8 (Lược giản về dạng chuẩn Chomsky). Với mọi văn phạm phi ngữ cảnh G, đều tồn tại văn phạm G_2 tương đương ở dạng chuẩn Chomsky.

Chứng minh: Xây dựng G_2 ở CNF tương đương với G.

Bước 1: Áp dụng định lý 5.6, 5.7 để *loại bỏ các qui tắc dư thừa*, các qui tắc rỗng hoá và qui tắc đơn vị. Giả sử văn phạm thu được là $G = (V_N, \Sigma, P, S)$.

Bước 2: Loại bỏ bớt các ký hiệu kết thúc ở vế phải chỉ để lại 1 ký hiệu. Định nghĩa $G_1 = (V_N', \Sigma, P_1, S)$ như sau:

+ Tất cả các qui tắc dạng $A \to a$ hoặc $A \to BC$ đưa vào P_1, tất cả các biến của V_N đều đưa vào V_N'.

+ Xét những qui tắc có dạng $A \to X_1X_2 \ldots X_n$, trong đó có một số phần tử là ký hiệu kết thúc. Giả sử $X_i = a_i$ là ký hiệu kết thúc, thì bổ sung thêm biến mới, ký hiệu là C_{ai} vào V_N' và bổ sung qui tắc $C_{ai} \to a_i$ vào P_1. Lặp lại quá trình trên cho tất cả các biến X_i nếu là các ký hiệu kết thúc.

Bước 3: Hạn chế bớt các biến ở vế phải. Theo các cách xây dựng như trên, P_1 là gồm các qui tắc có vế phải là một ký hiệu kết thúc (hoặc ký hiệu rỗng Λ) hoặc bằng hay nhiều hơn hai biến. Xây dựng $G_2 = (V_N'', \Sigma, P_2, S)$ như sau:

+ Tất cả các qui tắc của P_1 đưa vào P_2 nếu ở dạng CNF và tất cả các biến của V_N' đều đưa vào V_N''.

+ Xét những qui tắc có dạng $A \to A_1A_2 \ldots A_m$, trong đó $n \geq 3$. Tạo ra một số qui tắc mới $A \to A_1C_1$, $C_1 \to A_2C_2$, ... $C_{m-2} \to A_{m-1}A_m$ và bổ sung vào P_2, còn các biến mới $C_1, C_2, \ldots C_{m-2}$ bổ sung vào V_N''.

Dễ nhận thấy G_2 là ở dạng CNF tương đương với văn phạm G cho trước. ■

Văn phạm G_2 tương đương với G được gọi là dạng chuẩn CNF của G.

Ví dụ 5.8 Tìm dạng chuẩn CNF của văn phạm G có các qui tắc $S \to aAbB$, $A \to aA \mid a$, $B \to bB \mid b$.

Giải: G không có các qui tắc dư thừa, vậy có thể bỏ qua bước 1 và thực hiện từ bước 2.

Bước 2: Đặt $G_1 = (V_N', \Sigma, P_1, S)$, trong đó V_N', P_1 được xây dựng như sau:

+ $A \to a$, $B \to b$ đưa vào P_1.

+ $S \to aAbB$, $A \to aA$, $B \to bB$ chuyển thành các qui tắc

$S \to C_aAC_bB$, $A \to C_aA$, $B \to C_bB$, $C_a \to a$, $C_b \to b$ bổ sung vào P_1.

$V_N' = \{S, A, B, C_a, C_b\}$

Bước 3: Trong P_1 còn qui tắc $S \to C_aAC_bB$ cần phải chuyển về CNF. Qui tắc này được thay bằng: $S \to C_aC_1$, $C_1 \to AC_2$, $C_2 \to C_bB$.

Cuối cùng, dạng CNF của văn phạm trên sẽ là

$G_2 = (\{S, A, B, C_a, C_b, C_1, C_2\}, \{a, b\}, P_2, S)$, trong đó P_2 bao gồm:

$S \to C_aC_1$, $C_1 \to AC_2$, $C_2 \to C_bB$, $A \to C_aA$, $B \to C_bB$, $C_a \to a$, $C_b \to b$, $A \to a$, và $B \to b$

5.4.2 Dạng chuẩn Greibach

Một dạng chuẩn khác của văn phạm phi ngữ cảnh là dạng chuẩn Greibach cũng được sử dụng nhiều trong các chứng minh hay thiết kế các văn phạm.

Định nghĩa 5.9 Văn phạm phi ngữ cảnh G ở *dạng chuẩn Greibach* (viết tắt là GNF) nếu các qui tắc đều có dạng:

$A \to a\alpha$, trong đó $\alpha \in V_N^*$ và $a \in \Sigma$ (α có thể là Λ), và $S \to \Lambda$ nếu $\Lambda \in L(G)$. Khi Λ thuộc L(G) thì S không xuất hiện ở vế phải của mọi qui tắc.

Trong dạng chuẩn Greibach, vế phải của mọi qui tắc đều bắt đầu bằng một ký hiệu kết thúc.

Ví dụ: Văn phạm G có các qui tắc $S \to aAB \mid \Lambda$, $A \to bC$, $B \to b$, $C \to c$ là ở dạng GNF.

Vấn đề quan trọng đặt ra là có thể chuyển một văn phạm phi ngữ cảnh bất kỳ về GNF? Trước khi chứng minh định lý trả lời cho câu hỏi này chúng ta xét hai bổ đề bổ trợ sau.

Bổ đề 5.1 Giả sử $G = (V_N, \Sigma, P, S)$ là văn phạm phi ngữ cảnh. Cho trước $A \to B\gamma$ là A-dẫn xuất trong P và $B \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_k$. Đặt

$$P_1 = (P - \{A \to B\gamma\}) \cup \{A \to \beta_i\gamma \mid 1 \le i \le k\}$$

Khi đó $G_1 = (V_N, \Sigma, P_1, S)$ là văn phạm phi ngữ cảnh tương đương với G.

Chứng minh: Nếu áp dụng $A \to B\gamma$ trong dẫn xuất để $w \in L(G)$ thì chúng ta sẽ phải sử dụng $B \to \beta_i$ với i nào đó ở bước tiếp theo. Vậy $A \overset{*}{\underset{G}{\Rightarrow}} \beta_i\gamma$. Kết quả của việc sử dụng $A \to B\gamma$ và loại bỏ B khỏi P trong G thì hoàn toàn tương đương với việc sử dụng $A \to \beta_i\gamma$ trong G_1. Do vậy $L(G_1) = L(G)$. ■

Lưu ý: Bổ đề 5.1 được sử dụng để xoá bỏ biến B xuất hiện ở vị trí đầu tiên của các A- dẫn xuất.

Bổ đề 5.2 $G = (V_N, \Sigma, P, S)$ là văn phạm phi ngữ cảnh. Giả sử tập các A-dẫn xuất là $A \to A\alpha_1 \mid A\alpha_2 \mid \ldots A\alpha_n \mid \beta_1 \mid \beta_2 \ldots \mid \beta_m$, trong đó β_i không bắt đầu bằng A. Z là một biến mới. Đặt $G' = (V_N \cup \{Z\}, \Sigma, P_1, S)$ với P_1 được xác định như sau:

(a) Tập các A-dẫn xuất trong P_1 là $A \to \beta_1 \mid \beta_2 \ldots \mid \beta_m$, và

$$A \to \beta_1 Z \mid \beta_2 Z \ldots \mid \beta_m Z,$$

(b) Tập các Z-dẫn xuất trong P_1 là $Z \to \alpha_1 \mid \alpha_2 \mid \ldots \alpha_n$ và

$$Z \to \alpha_1 Z \mid \alpha_2 Z \mid \ldots \alpha_n Z$$

(c) Các qui tắc đối với các biến khác cũng thuộc P_1. Khi đó G′ là văn phạm phi ngữ cảnh tương đương với G.

Chứng minh:

Để chứng minh $L(G) \subseteq L(G')$ chúng ta chỉ cần lưu ý tới những xâu cuối được dẫn xuất bên trái nhất w trong G. Chỉ các qui tắc trong $P - P_1$ là $A \to A\alpha_1 \mid A\alpha_2 \mid \ldots A\alpha_n$

được sử dụng. Nếu $A \to A\alpha_i$ được sử dụng thì sau đó $A \to \beta_j$ nào đó cũng phải sử dụng để có được w. Nghĩa là $A \overset{*}{\underset{G'}{\Rightarrow}} w$.

Chứng minh chiều ngược lại $L(G') \subseteq L(G)$ tương tự. ■

Ví dụ 5.9 Áp dụng bổ đề 5.2 cho các A - dẫn xuất: $A \to aBD \mid bDB \mid c$, và $A \to AB \mid AD$

Giải:

Ở ví dụ này $\alpha_1 = B$, $\alpha_2 = D$, $\beta_1 = aBD$, $\beta_2 = bDB$, $\beta_3 = c$. Vậy các qui tắc mới là:

(a) $A \to aBD \mid bDB \mid c$, $A \to aBDZ \mid bDBZ \mid cZ$

(b) $Z \to B$, $Z \to D$, $Z \to BZ \mid DZ$

Lưu ý: Bổ đề 5.1 được sử dụng để loại bỏ biến A khỏi vế phải của các qui tắc $A \to A\alpha$.

Định lý 5.10 (Chuyển về dạng chuẩn Greibach). Mọi ngôn ngữ phi ngữ cảnh L đều có thể được sinh bởi văn phạm phi ngữ cảnh G ở dạng chuẩn Greibach.

Chứng minh: Chứng minh định lý theo hai trường hợp: L không chứa Λ và sau đó mở rộng khi có chứa Λ.

Trường hợp 1: $\Lambda \notin L(G)$

Bước 1: Loại bỏ các qui tắc rỗng và sau đó xây dựng văn phạm phi ngữ cảnh G ở dạng CNF để sinh ra L. Giả sử $G = (\{A_1, A_2, \ldots, A_n\}, \Sigma, P, A_1)$.

Bước 2: Để có được các qui tắc dạng $A_i \to a\gamma$ hoặc $A_i \to A_j\gamma$ với $j > i$, thì phải chuyển A_i -dẫn xuất về dạng $A_i \to A_j\gamma$ sao cho $j > i$. Điều này thực hiện được bằng phương pháp qui nạp theo i và sử dụng bổ đề 5.2. Cuối cùng chúng ta có $A_i \to A_j\gamma$, với $i = 1, 2, \ldots, n-1$ và $j > i$ hoặc $A_i \to a\gamma'$. A_n- dẫn xuất sẽ có dạng $A_n \to A_n\gamma$ hoặc $A_n \to a\gamma'$.

Bước 3: Chuyển A_n-dẫn xuất về dạng $A_n \to a\gamma$. Ở đây sử dụng bổ đề 5.2 để loại qui tắc dạng $A_n \to A_n\gamma$.

Bước 4: Chuyển trạng thái A_i-dẫn xuất về dạng $A_i \to a\gamma$ với $i = 1, 2, \ldots, n - 1$ (sử dụng bổ đề 5.1). Cuối bước 3 ta đã thu được $A_n \to a\gamma$. Các A_{n-1}-dẫn xuất có dạng $A_{n-1} \to A_n\gamma$ hoặc $A_{n-1} \to a\gamma'$. Áp dụng bổ đề 5.1 để loại đi những qui tắc $A_{n-1} \to A_n\gamma$ và chỉ còn lại những qui tắc dạng $A_{n-1} \to a\gamma$. Lặp lại tương tự đối với A_{n-2}, A_{n-3}, …, A_1.

Bước 5: Biến đổi các Z_i-dẫn xuất. Mỗi khi áp dụng bổ đề 5.2 ta sẽ có một biến mới, thêm biến Z_i khi áp dụng bổ đề với A_i-dẫn xuất. Khi đó Z_i-dẫn xuất có dạng $Z_i \to \alpha\gamma \mid A_k\gamma$, với k nào đó. Sau bước 4, vế phải của mọi A_k-dẫn xuất đều bắt đầu bằng một ký tự kết thúc. Ta áp dụng bổ đề 5.1 để loại bỏ $Z_i \to A_k\gamma$. Kết thúc bước 5 ta nhận được văn phạm ở dạng GNF.

Trường hợp 2: Xây dựng G để sinh ra L có chứa Λ. Trong trường hợp 1 chúng ta đã xây dựng được văn phạm phi ngữ cảnh $G = (V_N, \Sigma, P, S)$ để $L(G) = L - \Lambda$. Định nghĩa $G_1 = (V_N \cup \{S_1\}, \Sigma, P \cup \{S_1 \to S, S_1 \to \Lambda\}, S_1)$

Qui tắc $S_1 \to S$ có thể được loại bỏ theo định lý 5.7. Hiển nhiên $L(G_1) = L$. ■

Ví dụ 5.10 Tìm văn phạm phi ngữ cảnh ở dạng GNF tương đương với văn phạm G có các qui tắc: $S \to AA \mid a$, $A \to SS \mid b$.

Giải:

Bước 1: Văn phạm G ở dạng CNF. Đặt $A_1 = S$, $A_2 = A$ các qui tắc trên chuyển thành: $A_1 \to A_2A_2 \mid a$, $A_2 \to A_1A_1 \mid b$ và không có qui tắc rỗng.

Bước 2:

(i) A_1 - dẫn xuất có dạng đúng theo yêu cầu: $A_1 \to A_2A_2 \mid a$

(ii) $A_2 \to b$ là thoả mãn điều kiện bước 2 ở trên. Sử dụng bổ đề 5.1 để chuyển $A_2 \to A_1A_1$ về dạng: $A_2 \to A_2A_2A_1$, $A_2 \to aA_1$. Vậy A_2-dẫn xuất sẽ là: $A_2 \to A_2A_2A_1$, $A_2 \to aA_1$, $A_2 \to b$.

Bước 3: Áp dụng bổ đề 5.2 cho các A_2-dẫn xuất. Thêm Z_2 và chuyển $A_2 \to A_2A_2A_1$ về dạng yêu cầu:

$A_2 \to aA_1$, $A_2 \to b$, $A_2 \to aA_1Z_2$, $A_2 \to bZ_2$, $Z_2 \to A_2A_1$, $Z_2 \to A_2A_1Z_2$

Bước 4: (a) A_2 - dẫn xuất là $A_2 \to aA_1 \mid b \mid aA_1Z_2 \mid bZ_2$.

(b) Trong số các A_1-dẫn xuất chỉ còn lại $A_1 \to a$ vì $A_1 \to A_2A_2$ bị loại bỏ theo bổ đề 5.1. Các A_1-dẫn xuất sau khi chuyển đổi sẽ là:

$$A_1 \to a \mid aA_1A_2 \mid bA_2 \mid aA_1Z_2A_2 \mid bZ_2A_2.$$

Bước 5: Những Z_2- dẫn xuất cần biến đổi là: $Z_2 \to A_2A_1$, $Z_2 \to A_2A_1Z_2$

Áp dụng bổ đề 5.1 ta nhận được:

$Z_2 \to aA_1A_1 \mid bA_1 \mid aA_1Z_2A_1 \mid bZ_2A_2$,

$Z_2 \to aA_1A_1Z_2 \mid bA_1Z_2 \mid aA_1Z_2A_1Z_2 \mid bZ_2A_2Z_2$

Cuối cùng văn phạm cần tìm là: $G = (\{A_1, A_2, A_3\}, \{a, b\}, P_1, A_1\}$, trong đó P_1 có:

$A_1 \to a \mid aA_1A_2 \mid bA_2 \mid aA_1Z_2A_2 \mid bZ_2A_2$

$A_2 \to aA_1 \mid b \mid aA_1Z_2 \mid bZ_2$

$Z_2 \to aA_1A_1 \mid bA_1 \mid aA_1Z_2A_1 \mid bZ_2A_2$,

$Z_2 \to aA_1A_1Z_2 \mid bA_1Z_2 \mid aA_1Z_2A_1Z_2 \mid bZ_2A_2Z_2$

5.5 Bổ đề Bơm cho ngôn ngữ phi ngữ cảnh

Để chứng minh một ngôn ngữ cho trước là phi ngữ cảnh thì có nhiều cách. Một trong các cách đó là xây dựng một văn phạm phi ngữ cảnh (hay xây dựng ôtômát đẩy xuống sẽ được trình bày ở chương sau) để sinh ra (đoán nhận) ngôn ngữ đó.

Song, muốn chứng minh một ngôn ngữ không phải là phi ngữ cảnh thì khó hơn nhiều. Ta không thể xét được tất cả các văn phạm phi ngữ cảnh để chứng tỏ ngôn ngữ đã cho không được sinh bởi các văn phạm phi ngữ cảnh. Để khắc phục khó khăn này, người ta đưa ra một số "tiêu chuẩn đặc trưng" của ngôn ngữ phi ngữ cảnh và dựa vào đó để xem xét khả năng thỏa mãn của ngôn ngữ cho trước.

Sau đây chúng ta xét một số tính chất quan trọng của ngôn ngữ phi ngữ cảnh dựa vào bổ đề Bơm. Dựa vào đó chúng ta có thể khẳng định khá nhiều ngôn ngữ không phải là phi ngữ cảnh.

Bổ đề 5.3 Giả sử G là văn phạm phi ngữ cảnh ở dạng CNF và T là một cây dẫn xuất của G. Nếu độ dài của đường đi dài nhất trong T nhỏ hơn hoặc bằng *k* thì mọi kết quả sinh ra của T đều có độ dài nhỏ hơn hoặc bằng 2^{k-1}.

Chứng minh: Qui nạp theo độ dài k, độ dài của đường đi dài nhất trong số tất cả các *A-cây dẫn xuất* (cây dẫn xuất với gốc có nhãn là A).

Bước cơ sở qui nạp: Khi k = 1, tức là gốc của cây dẫn xuất chỉ có một đỉnh con mà nhãn của nó là ký hiệu kết thúc. Nếu đỉnh gốc có 2 đỉnh con thì nhãn của chúng sẽ phải là biến bởi G là ở dạng chuẩn Chomsky. Từ đó suy ra kết quả sinh ra có độ dài là $1 \le 2^{1-1} = 1$.

Bước giả thiết qui nạp: Giả sử kết luận trên đúng với mọi k – 1 (k > 1).

Ta cần chứng minh nó đúng với k, tức nếu T là A-cây dẫn xuất có các đường đi dài nhất nhỏ hơn hoặc bằng k thì mọi kết quả sinh ra của T đều có độ dài nhỏ hơn hoặc bằng 2^{k-1}. Bởi vì k > 1 nên đỉnh gốc của T có đúng hai cây con A_1 và A_2 có các đường đi dài nhất là nhỏ hơn hoặc bằng k – 1.

Theo giả thiết qui nạp mọi kết quả w_1, w_2 sinh ra trên hai cây dẫn xuất A_1 và A_2 tương ứng đều thỏa mãn $|w_1| \le 2^{k-2}$, $|w_2| \le 2^{k-2}$. Kết quả sinh ra bởi T khi đó có thể là w_1w_2 và $|w_1w_2| \le 2^{k-2} + 2^{k-2} = 2^{k-1}$. Đó là điều cần phải chứng minh. ■

Định lý 5.10 (**Bổ đề Bơm** cho ngôn ngữ phi ngữ cảnh). Nếu L là ngôn ngữ phi ngữ cảnh thì sẽ tồn tại số tự nhiên *n* sao cho:

(i) $\forall z \in L$ với $|z| \ge n$ đều có thể viết được dưới dạng: z = uvwxy, với u, v, x, y là các xâu nào đó

(ii) $|vx| \ge 1$

(iii) $|vwx| \le n$

(iv) $uv^kwx^ky \in L$ với mọi $k \ge 0$.

Chứng minh:

Từ hệ quả 5.2 ta có thể kiểm tra được $\Lambda \in L$ hay không. Khi $\Lambda \in L$ thì ta xây dựng văn phạm phi ngữ cảnh $G = (V_N, \Sigma, P, S)$ ở dạng chuẩn Chomsky để sinh ra $L - \{\Lambda\}$ (trường hợp $\Lambda \notin L$ thì xây dựng văn phạm $G = (V_N, \Sigma, P, S)$ ở dạng chuẩn Chomsky để sinh ra L).

Giả sử $|V_N| = m$ và đặt $n = 2^m$. Ta chỉ ra rằng n là số cần tìm ở bổ đề trên.

Với $z \in L$, $|z| \geq n$ ta đi xây dựng cây dẫn xuất cho z. Nếu độ dài đường đi dài nhất trong T nhiều nhất là m thì theo bổ đề 5.3, ta có $|z| \leq 2^{m-1}$. Nhưng $|z| \geq n = 2^m > 2^{m-1}$. Vậy T sẽ có đường đi, ví dụ Γ, có độ dài lớn hơn hoặc bằng m + 1. Γ có ít nhất m + 2 đỉnh và chỉ có đỉnh cuối cùng là lá. Như thế trên Γ sẽ có m + 1 đỉnh có nhãn là các biến. Bởi vì $|V_N| = m$ nên một số nhãn sẽ phải lặp lại.

Chúng ta chọn đỉnh có nhãn bị lặp lại như sau: bắt đầu từ lá đi ngược lên đến khi gặp đỉnh đầu tiên có nhãn bị lặp lại, ví dụ đó là nhãn B. Giả sử v_1 và v_2 là hai đỉnh có cùng nhãn B, với v_1 ở gần gốc của cây T hơn. Trên đoạn đường đi Γ từ v_1 đến lá chỉ chứa đúng một đỉnh có nhãn B bị lặp 1 lần, do vậy độ dài lớn nhất của đoạn đó là m + 1.

Giả sử T_1 và T_2 là hai cây con tương ứng với hai đỉnh gốc v_1 và v_2, T_1 sinh ra z_1 và T_2 sinh ra w. Theo bổ đề 5.3 suy ra $|z_1| \leq 2^m$.

Bởi vì z là xâu được sinh ra bởi cây T và T_1 là cây con thực sự của T, nên có thể viết $z = uz_1y$. Mặt khác, T_2 lại là cây con thực sự của T_1, nên ta viết được $z_1 = vwx$. $|vwx| > |w|$ suy ra $|vx| \geq 1$. Do vậy, $z = uvwxy$ với $|vwx| \leq n$ và $|vx| \geq 1$. Đến đây ta đã chứng minh được các điều (i) – (iii).

Bởi T là S-cây dẫn xuất và T_1, T_2 là hai B-cây dẫn xuất, nghĩa là $S \overset{*}{\Rightarrow} uBy$, $B \overset{*}{\Rightarrow} vBx$ và $B \overset{*}{\Rightarrow} w$. Vì $S \overset{*}{\Rightarrow} uBy \Rightarrow uwy = uv^0wx^0y \in L$. Với $k \geq 1$, $S \overset{*}{\Rightarrow} uBy \overset{*}{\Rightarrow} uv^kBx^ky \overset{*}{\Rightarrow} uv^kwx^ky \in L$. Điều này chứng minh điều kiên (iv) của bổ đề Bơm. ■

Ví dụ 5.11 Chỉ ra rằng $L = \{a^p \mid p \text{ là số nguyên tố}\}$ không phải là ngôn ngữ phi ngữ cảnh.

Giải: Giả sử L là phi ngữ cảnh. Theo bổ đề Bơm thì sẽ tìm được số *n* để thoả mãn các điều kiện của định lý 5.10.

$z = a^p$, p là số nguyên tố và $p \geq n$. Ta có thể viết nó dưới dạng $z = uvwxy$. Chọn k = 0, theo bổ đề Bơm ta có $uv^0wx^0y \in L$. Suy ra $|uwy| = q$ là số nguyên tố. Đặt $|vx| = m$ và xét tiếp dãy $uv^qwx^qy \in L$. Nó có độ dài $|uv^qwx^qy| = q + q * m = q*(1 + m)$, lại không phải là số nguyên tố, do đó mâu thuẫn với giả thiết.

Bổ đề Bơm được sử dụng để chỉ ra một ngôn ngữ không phải là phi ngữ cảnh. Trước tiên, giả sử L là ngôn ngữ phi ngữ cảnh. Áp dụng bổ đề Bơm thì dẫn đến mâu thuẫn.

Thuật toán áp dụng bổ đề Bơm để kiểm tra xem L có phải là phi ngữ cảnh không.

1. Giả sử L là phi ngữ cảnh và *n* là số tự nhiên nhận được từ bổ đề Bơm.
2. Chọn $z \in L$ sao $|z| \geq n$. Viết $z = uvwxy$ theo bổ đề.
3. Tìm một số *k* sao cho $uv^kwx^ky \notin L$. Điều này là mâu thuẫn, do vậy L không phải là phi ngữ cảnh.

Ví dụ 5.12 Hãy chỉ ra rằng L = $\{a^nb^nc^n \mid n \geq 1\}$ không phải là ngôn ngữ phi ngữ cảnh mà là ngôn ngữ cảm ngữ cảnh.

Giải:

Bước 1. Giả sử L là phi ngữ cảnh va *n* là số nguyên như trong bổ đề Bơm nêu trên.

Bước 2. Chọn z = $a^nb^nc^n$. Khi đó |z| = 3n > n. Theo bổ đề ta viết được z = uvwxy, trong đó |vx| > 1, suy ra ít nhất là *v* hoặc *x* sẽ khác rỗng.

Bước 3. uvwxy = $a^nb^nc^n$. Vì $|uwx| \leq n$ nên suy ra $1 \leq |ux| \leq n$. Dễ dàng thấy rằng, v và x không thể chứa tất cả ba chữ a, b, c. Khi đó ta có

1. v hoặc x có dạng a^ib^j (hoặc b^ic^j) với $i + j \leq n$, $i, j \geq 1$.
2. v hoặc x chỉ chứa một trong ba chữ a, b, c.

Trường hợp 1. Nếu v = a^ib^j (hoặc x = a^ib^j) thì $v^2 = a^ib^j\, a^ib^j$ (tương tự $x^2 = a^ib^ja^ib^j$) với $i + j \leq n$. Bởi vì v^2, x^2 là hai xâu con của uv^2wx^2y, nên $ua^ib^ja^ib^jwa^ib^j\, a^ib^jy$ không thể viết thành dạng $a^mb^mc^m$ được, nghĩa là $uv^2wx^2y \notin L$. Tương tự đối với trường hợp v = b^ic^j (hoặc x = b^ic^j).

Trường hợp 2. Nếu *v, x* chỉ chứa một trong ba chữ a, b, c (ví dụ v = a^i và x = b^j, hoặc v = b^i và x = c^j với $1 \leq i, j \leq n$), thì xâu uwy sẽ chứa chữ thứ ba còn lại, gọi là a_1. Hiển nhiên a_1^n sẽ là xâu con của uwy vì a_1 không xuất hiện trong *v* và *x*. Biết rằng uvwxy = $a^nb^nc^n$ nên số các hai chữ khác với a_1 trong uwy là nhỏ hơn n, nghĩa là $uwy \notin L$. Mặt khác $uv^0wx^0y = uwy \notin L$.

Vậy L không phải là phi ngữ cảnh.

Trong ví dụ 3.10 chúng ta đã xây dựng văn phạm cảm ngữ cảnh để đoán nhận L.

5.6 Thuật toán quyết định được đối với các ngôn ngữ phi ngữ cảnh

Các tính chất nêu trên giúp chúng ta xây dựng được các thuật toán giải quyết những vấn đề sau:

(i) *Ngôn ngữ phi ngữ cảnh L có rỗng hay không?.* Theo định lý 5.3 ta có thể xác định $V_N' = W_k$ và L khác rỗng $\Leftrightarrow S \in W_k$.

(ii) *Ngôn ngữ phi ngữ cảnh L có hữu hạn hay không?.* Xây dựng văn phạm phi ngữ cảnh không dư thừa G ở CNF để sinh ra L - {Λ}. Xây dựng đồ thị định hướng tương ứng với G. Nếu $A \rightarrow BC$ thuộc P thì có hai cung định hướng đi từ A đến B và từ A đến C trong đồ thị đó. L là hữu hạn khi và chỉ khi đồ thị tương ứng của văn phạm G là phi chu trình.

(iii) *Ngôn ngữ chính qui L có rỗng hay không?.* Xây dựng ôtômát đơn định hữu hạn đoán nhận được L. Xác định tập tất cả các trạng thái đạt được từ trạng thái đầu q_0. Tìm các trạng thái đạt được từ q_0 qua một lần áp dụng ký hiệu vào. Tất cả các trạng thái này được sắp xếp theo hàng tương ứng với ký hiệu vào. Công việc trên kết thúc sau hữu hạn bước. Nếu có trạng thái kết thúc xuất hiện trong bảng trên thì L là khác rỗng, ngược lại là ngôn ngữ rỗng.

(iv) *Ngôn ngữ chính qui L có vô hạn hay không?*. Xây dựng ôtômát đơn định hữu hạn M đoán nhận được L. L là vô hạn khi và chỉ khi M có chu trình.

Bài tập về ngôn ngữ phi cảnh

5.1 Cho trước văn phạm G có các qui tắc: S → S + S | S * S, S → a | b. Tìm cây dẫn xuất cho kết quả a * b + a * b thuộc L(G).

5.2 Cho trước văn phạm G có các qui tắc: S → 0S0 | 1S1 | A, A → 2B3, B→2B3 | 3. Tìm L(G)?

5.3 Cho trước văn phạm G có các qui tắc: S → aB | bA, A → aS | bAA | a, B→bS | aBB | b. Đối với xâu w = aaabbabbba, hãy tìm:

(i) Dãy dẫn xuất bên trái nhất của w,

(ii) Dãy dẫn xuất bên phải nhất của w,

(iii) Cây dẫn xuất của w.

5.4 Chỉ ra rằng văn phạm G có các qui tắc S → SbS | a là nhập nhằng.

5.5 Chỉ ra rằng văn phạm G có các qui tắc S → a | abS | aAb, A → bS | aAAb là nhập nhằng.

5.6 Xét văn phạm có các qui tắc sau: S → aB | bA, A → aS | bAA | a, B → bS | aBB | b. Đối với từ w = aaabbabbba, hay tìm

(i) Dẫn xuất trái nhất của w

(ii) Dẫn xuất phải nhất của w

(iii) Cây phân tích cú pháp của w.

5.7 Xây dựng văn phạm không dư thừa tương đương với văn phạm G có các qui tắc sau:

(i) S → aAa, A → Sb | bCC | DaA, C → abb | DD, E → aC, D → aDA

(ii) S → aAa, A → bBB, B → ab, C → aB

(iii) S → AB | CA, B → BC | AB, A → a, C → aB | b

5.8 Tìm văn phạm tối giản tương đương với văn phạm có các qui tắc:

(i) S → 0S0 | 1S1 | A, A → 2B3, B→2B3 | 3

(ii) S → aAa, A → Sb | bCC | DaA, C → abb | DD, E → aC, D → aDA

(iii) S → a | abS | aAb, A → bS | aAAb

5.9 Chuyển các văn phạm có các qui tắc sau về dạng CNF:

(i) S → a | b | cSS

(ii) S → 1A | 0B, A → 1AA | 0S | 0, B → 0BB | 1S | 1

(iii) S → abSb | a | aAb, A → bS | aAAb

5.10 Tìm dạng chuẩn Chomsky của văn phạm có các qui tắc sau: $S \to aAD$, $A \to aB \mid bAB$, $B \to b$, $D \to d$.

5.11 Tìm dạng chuẩn Greibach của văn phạm có các qui tắc sau: $S \to AB$, $A \to BS \mid b$, $B \to SA \mid a$.

5.12 Chuyển các văn phạm có các qui tắc sau về dạng GNF:

(i) $S \to SS$, $S \to 0S1 \mid 01$

(ii) $S \to AB$, $A \to BSB$, $A \to BB$, $B \to aBb$, $B \to a$, $A \to b$

(iii) $S \to A0$, $A \to 0B$, $B \to A0$, $B \to 1$

5.13 Nếu $w \in L(G)$ và $|w| = k$, anh (chị) có thể nói gì về số bước dẫn xuất để dẫn ra w, khi

(i) G là CNF

(ii) G là GNF

5.14 Hãy chỉ ra rằng những ngôn ngữ sau không phải là ngôn ngữ phi ngữ cảnh.

(i) $L = \{a^n b^n c^n \mid n \geq 1\}$

(ii) $L = \{a^m b^n \mid n = m^2\}$

(iii) $L = \{a^m b^m c^n \mid m \leq n \leq 2m\}$

(iv) $L = \{a^{n^2} \mid n \geq 1\}$

5.15 Một văn phạm phi ngữ cảnh được gọi là tự nhúng nếu tồn tại biến A sao cho $A \overset{*}{\Rightarrow} uAv$, trong đó $u, v \in \Sigma^*$, $u, v \neq \Lambda$. Hãy chỉ ra rằng một ngôn ngữ phi ngữ cảnh là chính qui khi và chỉ khi nó được sinh ra bởi một văn phạm không tự nhúng.

5.16 Chỉ ra rằng mọi ngôn ngữ phi ngữ cảnh không chứa Λ đều được sinh ra bởi một văn phạm phi ngữ cảnh có các qui tắc dạng $A \to a \mid a\alpha b$.

CHƯƠNG VI
Ôtômát đẩy xuống

Chương này đề cập đến:

- ✓ Ôtômát đẩy xuống,
- ✓ Hai kiểu đoán nhận của ôtômát đẩy xuống,
- ✓ Chứng minh rằng tập các kết quả đoán nhận của ôtômát đẩy xuống chính là lớp các ngôn ngữ phi ngữ cảnh.

6.1 Các định nghĩa cơ sở

Mô hình toán học thực sự hữu hạn được nghiên cứu trong các chương trước vẫn còn chưa thoả đáng với phần lớn các mục tiêu của các lĩnh vực khoa học, đặc biệt là khoa học máy tính. Nó không có khả năng đoán nhận ngay cả một ngôn ngữ đơn giản như $L = \{a^n b^n \mid n \geq 1\}$ vì nó không thể đếm được số chữ a đọc được trước khi chữ b đầu tiên được tính đến. Không thể xây dựng ôtômát hữu hạn trạng thái để đoán nhận được L bởi vì các xâu của nó có dạng $a^n b^n$ với n là tuỳ ý và để sinh ra chúng thì phải có vô hạn trạng thái. Tuy nhiên, việc mở rộng trực tiếp ôtômát hữu hạn bằng cách cho phép nó có vô số các trạng thái rõ ràng là không phải là mô hình tính toán phù hợp. Nó quá tổng quát và không thoả mãn yêu cầu chủ yếu về tính hiệu quả của khoa học tính toán.

Vậy ôtômát loại nào thích hợp để đoán nhận được những ngôn ngữ như thế?. Câu trả lời chính là *ôtômát đẩy xuống*.

Một cách tự nhiên để tìm kiến thông tin từ băng làm việc dựa trên nguyên tắc ngăn xếp "vào trước – ra sau" (hay "vào sau – ra trước"). Một băng làm việc như vậy được nói đến như một *băng (kho) đẩy xuống*. Một ôtômát đẩy xuống là một ôtômát hữu hạn kèm theo một băng đẩy xuống vô hạn tiềm năng.

Ôtômát đẩy xuống hoạt động như sau: Nó đọc bảng chữ (ký hiệu) vào, có trạng thái điều khiển, trạng thái bắt đầu, kết thúc giống như ôtômát hữu hạn. Ngoài ra, nó còn có kho (bộ nhớ) đẩy xuống. Mỗi lần thay đổi trạng thái nó có thể đọc, lấy ra từ đỉnh xuống đáy hoặc ghi một ký hiệu vào kho đẩy xuống. Ôtômát đẩy xuống được định nghĩa hình thức như sau.

Định nghĩa 6.1 *Ôtômát đẩy xuống* (PDA) là bộ bảy:

$A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, trong đó

(i) Q là tập hữu hạn khác rỗng các trạng thái,

(ii) Σ là tập hữu hạn khác rỗng các ký hiệu vào, bảng chữ cái,

(iii) Γ là tập hữu hạn khác rỗng các ký hiệu đẩy xuống trong ngăn xếp,

(vi) q_0 là trạng thái bắt đầu, $q_0 \in Q$,

(vii) $Z_0 \in \Gamma$ ký hiệu đẩy xuống đặc biệt được gọi là ký hiệu khởi đầu,

(viii) $F \subseteq Q$ là tập các trạng thái kết thúc,

(ix) $\delta: Q \times (\Sigma \cup \{\Lambda\}) \times \Gamma \rightarrow Q \times \Gamma^*$.

Ví dụ 6.1 $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, trong đó $Q = \{q_0, q_1, q_f\}$, $\Sigma = \{a, b\}$,

$\Gamma = \{a, Z_0\}$, $F = \{q_f\}$ và δ được xác định như sau:

$\delta(q_0, a, Z_0) = \{(q_0, aZ_0)\}$, $\delta(q_1, b, a) = \{(q_1, \Lambda)\}$

$\delta(q_0, a, a) = \{(q_0, aa)\}$, $\delta(q_1, \Lambda, Z_0) = \{(q_1, \Lambda)\}$

$\delta(q_0, b, a) = \{(q_1, \Lambda)\}$

Lưu ý: 1. Ta có thể viết $\delta(q, a, Z) \subseteq Q \times \Gamma^*$ là tập con hữu hạn, có thể là tập rỗng, các ký hiệu đẩy xuống được lưu trữ trong ngăn xếp hay bộ nhớ đẩy xuống (PDS).

2. Ở mọi thời điểm, PDA ở trạng thái q đọc ký hiệu vào *a* và ký hiệu đẩy xuống *Z* ở đỉnh (đầu ngăn xếp). Hàm δ chuyển từ trạng thái *q* sang q′ và viết xâu α sau khi rời khỏi Z, tức là $(q', \alpha) \in \delta(q, a, Z)$. Khi $\alpha = \Lambda$ thì ký hiệu trên đỉnh bị xoá đi.

3. Hành vi của PDA là không đơn định vì hàm chuyển trạng thái $\delta(q, a, Z)$ là tập con, trong đó phần tử tương ứng với một cặp trạng thái và một dãy ký hiệu ở kho đẩy xuống PDS.

4. δ xác định trên $Q \times (\Sigma \cup \{\Lambda\}) \times \Gamma$, PDA có thể vẫn chuyển được sang trạng thái khác mà không đọc ký hiệu vào nào ($\delta(q, a, Z) = \phi$). Những chuyển trạng thái như thế được gọi là Λ- dịch chuyển.

5. PDA sẽ không dịch chuyển được khi PDS là tập rỗng.

6. Khi viết $\alpha = Z_1Z_2 \ldots Z_m$ ở PDS thì Z_1 là ở đỉnh, dưới đó là Z_2, …

Định nghĩa 6.2 $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ là PDA. Một hình trạng hay một mô tả hiện thời ID là bộ (q, x, α), trong đó $q \in Q$, $x \in \Sigma^*$, $\alpha \in \Gamma^*$.

Ví dụ: $(q, a_1a_2\ldots a_n, Z_1Z_2 \ldots Z_m)$ là ID. Nó mô tả hoạt động của PDA ở trạng thái hiện thời q, đọc dãy các ký hiệu vào $a_1a_2\ldots a_n$ theo thứ tự từ trái qua phải. PDS của nó là $Z_1Z_2 \ldots Z_m$, trong đó Z_1 là phần tử ở đỉnh, Z_2 là phần tử thứ hai, …

Định nghĩa 6.3 Hình trạng bắt đầu là (q_0, x, Z_0), nghĩa là PDA luôn bắt đầu từ trạng thái q_0, xử lý xâu vào *x* khi ở PDS chỉ có một ký hiệu khởi đầu Z_0.

Lưu ý: Đối với ôtômát hữu hạn, có thể mô tả dòng công việc bằng dãy chuyển đổi trạng thái. Trong trường hợp PDA, còn có thêm cấu trúc kho đẩy xuống, do vậy để mô tả hoạt động của PDA phải sử dụng dãy chuyển trạng thái các hình trạng ID.

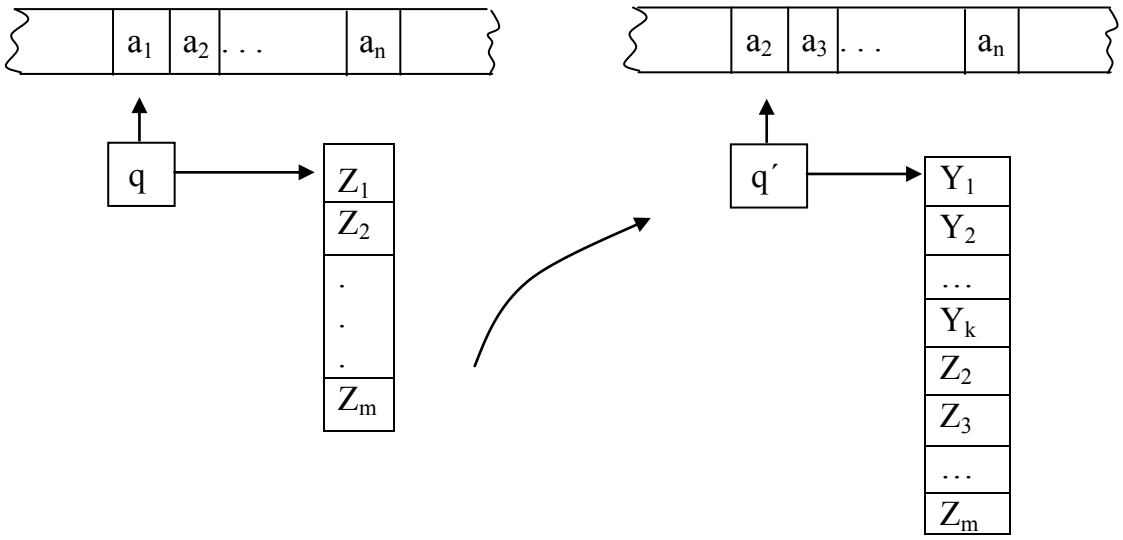
Định nghĩa 6.4 Cho A là một PDA. Quan hệ chuyển đổi giữa các hình trạng ID (ký hiệu là $\vdash$) được định nghĩa như sau:

$$(q, a_1a_2\ldots a_n, Z_1Z_2\ldots Z_m) \vdash (q', a_2\ldots a_n, \beta Z_2\ldots Z_m) \qquad (6.1)$$

nếu $\delta(q, a_1, Z_1)$ chứa (q', β). Quan hệ chuyển đổi các hình trạng sẽ tạo thành *dãy biến đổi hình trạng.*

PDA ở trạng thái q với các ký hiệu đẩy xuống $Z_1Z_2\ldots Z_m$ ở PDS đọc ký hiệu vào a_1. Sau chuyển đổi hình trạng trên thì xâu vào cần xử lý là $a_2\ldots a_n$.

Nếu $\beta = Y_1Y_2\ldots Y_k$ thì quan hệ (6.1) có thể mô tả như sau:



Hình H6-1 Mô tả quan hệ chuyển trạng thái của PDA

Lưu ý: Chúng ta có thể định nghĩa bao đóng bắc cầu của quan hệ $\vdash$ là $\vdash^*$ như sau:

$(q, x, \alpha) \vdash^* (q', y, \beta)$ nếu tồn tại $n \geq 0$, $q_1, q_2, \ldots, \in Q$; $x_1, x_2, \ldots \in \Sigma^*$, $\alpha_1, \alpha_2, \ldots \in \Gamma^*$ để $(q, x, \alpha) \vdash (q_1, x_1, \alpha_1) \vdash (q_2, x_2, \alpha_2) \ldots \vdash (q', y, \beta)$.

Hai tính chất sau thường được sử dụng trong kiến thiết các PDA và chứng minh các định lý quan trọng về ôtômat đẩy xuống.

Tính chất 6.1 Nếu $(q_1, x, \alpha) \vdash^* (q_2, \Lambda, \beta)$ thì với mọi $y \in \Sigma^*$

$$(q_1, xy, \alpha) \vdash^* (q_2, y, \beta) \qquad (6.2)$$

Ngược lại, nếu $(q_1, xy, \alpha) \vdash^* (q_2, y, \beta)$ thì $(q_1, x, \alpha) \vdash^* (q_2, \Lambda, \beta)$

Chứng minh: Nếu PDA ở trạng thái q_1 với dãy các ký hiệu đẩy xuống α, nó chuyển xuống dưới lần lượt sau khi xử lý xong dãy ký hiệu vào *x* để chuyển về trạng thái q_2 với dãy ký hiệu đẩy xuống β. PDA sẽ hoạt động tương tự khi nhận đầu vào *xy*, nhưng chỉ xử lý *x*, phần *y* sẽ vẫn giữ lại như hình H6-1. ■

Tính chất 6.2 Nếu $(q, x, \alpha) \vdash^* (q', \Lambda, \gamma)$ (6.3)

thì với mọi $\beta \in \Gamma^*$, $(q, x, \alpha\beta) \vdash^* (q', \Lambda, \gamma\beta)$ (6.4)

Chứng minh: Theo định nghĩa, dãy (6.3) được viết như sau:

$$(q, x, \alpha) \vdash (q_1, x_1, \alpha_1) \vdash (q_2, x_2, \alpha_2) \vdash \ldots \vdash (q', \Lambda, \gamma)$$

Xét $(q_i, x_i, \alpha_i) \vdash (q_{i+1}, x_{i+1}, \alpha_{i+1})$. Đặt $\alpha_i = Z_1Z_2\ldots Z_m$. Khi chuyển trạng thái, Z_1 bị xoá đi và một xâu nào đó được thay vào trước $Z_2\ldots Z_m$. Như vậy, $Z_2\ldots Z_m$ không bị ảnh hưởng. Nếu ta có β đứng sau $Z_2\ldots Z_m$ thì $Z_2\ldots Z_m\beta$ cũng không bị thay đổi. Từ đó suy ra $(q_i, x_i, \alpha_i\beta) \vdash (q_{i+1}, x_{i+1}, \alpha_{i+1}\beta)$, nghĩa là

$$(q, x, \alpha\beta) \vdash (q_1, x_1, \alpha_1\beta) \vdash (q_2, x_2, \alpha_2\beta) \vdash \ldots \vdash (q', \Lambda, \gamma\beta) \qquad \blacksquare$$

Lưu ý: (6.4) không suy ra (6.3). Xét trường hợp:

$A = (\{q_0\}, \{a, b\}, \{Z_0\}, \delta, q_0, Z_0, \varnothing)$ với

$\delta(q_0, a, Z_0) = \{(q_0, \Lambda)\}$, $\delta(q_0, b, Z_0) = \{(q_0, Z_0Z_0)\}$

$$\begin{aligned}(q_0, aab, Z_0Z_0Z_0Z_0) &\vdash (q_0, ab, Z_0Z_0Z_0)\\ &\vdash (q_0, ab, Z_0Z_0Z_0)\\ &\vdash (q_0, b, Z_0Z_0)\\ &\vdash (q_0, \Lambda, Z_0Z_0Z_0)\end{aligned}$$

Nghĩa là $(q_0, aab, Z_0Z_0Z_0Z_0) \vdash^* (q_0, \Lambda, Z_0Z_0Z_0)$. Đặt $\alpha = Z_0Z_0Z_0$, $\beta = Z_0$, $\gamma = Z_0Z_0$ thì (6.3) suy ra (6.4), nhưng ngược lại thì không dẫn ra được bởi vì

$(q_0, aab, Z_0) \vdash (q_0, ab, \Lambda)$ và sau đó không chuyển trạng thái được tiếp vì PDS là rỗng.

Định nghĩa 6.5 PDA $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ là đơn định nếu

(a) $\delta(q, a, Z)$ hoặc là tập rỗng hoặc chỉ có một phần tử,

(b) $\delta(q, \Lambda, Z) \neq \varnothing$ thì $\delta(q, a, Z) = \varnothing$, $\forall a \in \Sigma$

6.2 Các kết quả đoán nhận bởi PDA

Định nghĩa 6.6 Cho $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ là một PDA. Tập đoán nhận được bởi A với trạng thái kết thúc (được A đoán nhận với trạng thái kết thúc), ký hiệu T(A), được định nghĩa như sau:

$$T(A) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q_f, \Lambda, \alpha) \text{ với } q_f \in F \text{ và } \alpha \in \Gamma^*\}$$

Ví dụ 6.2 Cho $A = (\{q_0, q_1, q_f\}, \{a, b, c\}, \{a, b, Z_0\}, \delta, q_0, Z_0, \{q_f\})$, trong đó

$\delta(q_0, a, Z_0) = \{(q_0, aZ_0)\}$, $\delta(q_0, b, Z_0) = \{(q_0, bZ_0)\}$, (6.5)

$\delta(q_0, a, a) = \{(q_0, aa)\}$, $\delta(q_0, b, a) = \{(q_0, ba)\}$, (6.6)

$\delta(q_0, a, b) = \{(q_0, ab)\}$, $\delta(q_0, b, b) = \{(q_0, bb)\}$, (6.7)

$\delta(q_0, c, a) = \{(q_1, a)\}$, $\delta(q_0, c, b) = \{(q_1, b)\}$, (6.8)

$\delta(q_0, c, Z_0) = \{(q_1, Z_0)\}$

$\delta(q_1, a, a) = \delta(q_1, b, b) = \{(q_1, \Lambda)\}$ (6.9)

$\delta(q_1, \Lambda, Z_0) = \{(q_f, Z_0)\}$ (6.10)

Trước tiên chúng ta xét hình trạng đầu $(q_0, bacab, Z_0)$

$(q_0, bacab, Z_0) \vdash (q_0, acabb, bZ_0)$ theo (6.5)

$\vdash (q_0, cab, abZ_0)$ theo (6.7)

$\vdash (q_1, ab, abZ_0)$ theo (6.8)

$\vdash (q_1, b, bZ_0)$ theo (6.9)

$\vdash (q_1, \Lambda, Z_0)$ theo (6.10)

$\vdash (q_f, \Lambda, Z_0)$ theo (6.10)

Nghĩa là $(q_0, bacab, Z_0) \vdash^* (q_f, \Lambda, Z_0)$, do đó bacab $\in T(A)$.

Tương tự chúng ta có thể chứng minh được rằng $wcw^T \in T(A)$.

Vậy $L = \{wcw^T \mid w \in \{a, b\}^*\} \subseteq T(A)$.

Để chứng chiều ngược lại, $T(A) \subseteq L$ ta chỉ cần chỉ ra rằng $L^C \subseteq T(A)^C$ (phần bù của chúng cũng bao nhau). Giả sử $x \in L^C$. Xét tiếp hai trường hợp:

Trường hợp 1: x không chứa ký tự c. Khi đó PDA A không thể chuyển đổi từ trạng thái q_0 đến q_1 và để đến q_f được. Vậy $x \in T(A)^C$.

Trường hợp 1: x có chứa ký tự c, $x = w_1cw_2$, $w_1^T \neq w_2$.

$(q_0, w_1cw_2, Z_0) \vdash^* (q_0, cw_2, w_1^TZ_0) \vdash (q_1, w_2, w_1^TZ_0)$

Bởi vì $w_1^T \neq w_2$ nên $(q_1, w_2, w_1^TZ_0)$ không thể chuyển về được (q_1, Λ, Z_0) để kết thúc ở (q_f, Λ, Z_0). Từ đó cũng suy ra $x \in T(A)^C$.

Kết luận $T(A) = L$.

Định nghĩa 6.7 Cho $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ là một PDA. Tập đoán nhận được bởi một kho rỗng của A (được A đoán nhận với kho đẩy xuống là rỗng), ký hiệu N(A), được định nghĩa như sau:

$N(A) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q, \Lambda, \Lambda)$ với $q \in Q\}$

Ví dụ 6.3 Xét tiếp PDA A được cho ở ví dụ 6.2. Nếu chúng ta bổ sung thêm

$\delta(q_f, \Lambda, Z_0) = \{(q_f, \Lambda)\}$ thì

$N(A) = \{wcw^T \mid w \in \{a, b\}^*\}$

Định lý 6.1 Nếu $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ là một PDA đoán nhận L bằng kho đẩy xuống rỗng thì sẽ tìm được PDA $B = (Q', \Sigma, \Gamma', \delta', q_0', Z_0', F')$ đoán nhận L bằng trạng thái kết thúc, nghĩa là $L = N(A) = T(B)$.

Chứng minh: Xây dựng B sao cho:

(a) Chuyển động bắt đầu của B đạt được từ hình trạng khởi đầu của A,

(b) Chuyển động kết thúc của B đạt được tương ứng đến trạng thái kết thúc của A,

(c) Mọi chuyển động trung gian trong B giống như trong A.

PDA B được xây dựng như sau:

$B = (Q', \Sigma, \Gamma', \delta', q_0', Z_0', F')$, trong đó

q_0' là trạng thái mới không có trong Q

$F' = \{q_f\}$, q_f cũng là trạng thái mới không thuộc Q

$Q' = Q \cup \{q_0', q_f\}$,

Z_0' là ký hiệu khởi đầu mới của PDS,

$\Gamma' = \Gamma \cup \{Z_0'\}$ và

δ' được cho theo ba qui tắc:

R1: $\delta'(q_0', \Lambda, Z_0') = \{(q_0, Z_0Z_0')\}$

R2: $\delta'(q, a, Z) = \delta(q, a, Z)$ với mọi $(q, a, Z) \in Q \times (\Sigma \cup \{\Lambda\} \times \Gamma$

R3: $\delta'(q, \Lambda, Z_0') = \{(q_f, \Lambda)\}$ với mọi $q \in Q$

Chứng minh chi tiết tham khảo tài liệu [4].

Theo cách thiết kế như trên thì B là đơn định khi và chỉ khi A đơn định. ■

Ví dụ 6.4 Xét PDA $A = (\{q_0, q_1\}, \{a, b\}, \{a, Z_0\}, \delta, q_0, Z_0, \varnothing)$, với δ cho trước như sau:

R1: $\delta(q_0, a, Z_0) = \{(q_0,aZ_0)\}$

R2: $\delta(q_0, a, a) = \{(q_0, aa)\}$

R3: $\delta(q_0, b, a) = \{(q_1, \Lambda)\}$

R4: $\delta(q_1, b, a) = \{(q_1, \Lambda)\}$

R5: $\delta(q_1, \Lambda, a) = \{(q_1, \Lambda)\}$

Xác định N(A) và kiến thiết B để T(B) = N(A).

Giải: Trước tiên chúng ta có thể chỉ ra $N(A) = \{a^nb^n \mid n \geq 1\}$. Thật vậy

$(q_0, a^nb^n, Z_0) \vdash^* (q_0, b^n, a^nZ_0)$ bằng cách áp dụng R1 và R2

$\vdash^* (q_1, \Lambda, Z_0)$ bằng cách áp dụng R3 và R4

$\vdash (q_1, \Lambda, \Lambda)$ bằng cách áp dụng R5

Suy ra $a^nb^n \in N(A)$. Chiều ngược lại được dẫn ra từ các tính chất của các qui tắc R1-R5.

Tiếp theo chúng ta xây dựng PDA $B = (Q', \{a, b\}, \Gamma', \delta', q_0', Z_0', F')$, trong đó

$Q' = \{q_0, q_0', q_1, q_f\}$, $\Gamma' = \{a, b, Z_0'\}$, $F' = \{q_f\}$ và

$\delta'(q_0', \Lambda, Z_0') = \{(q_0, Z_0'Z_0')\}$ $\quad$ $\delta'(q_0, a, Z_0) = \{(q_0, aZ_0)\}$

$\delta'(q_0, a, a) = \{(q_0, aa)\}$ $\quad$ $\delta'(q_0, b, a) = \{(q_1, \Lambda)\}$

$\delta'(q_1, b, a) = \{(q_1, \Lambda)\}$ $\quad$ $\delta'(q_1, \Lambda, Z_0') = \{(q_1, \Lambda)\}$

$\delta'(q_0, \Lambda, Z_0') = \{(q_f, \Lambda)\}$ $\quad$ $\delta'(q_1, \Lambda, Z_0') = \{(q_f, \Lambda)\}$

Kết luận $\quad T(B) = N(A) = \{a^nb^n \mid n \geq 1\}$.

Định lý 6.2 Nếu $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ là một PDA đoán nhận L bằng trạng thái kết thúc thì sẽ tìm được PDA $B = (Q', \Sigma, \Gamma', \delta', q_0', Z_0', F')$ đoán nhận L bằng kho đẩy xuống rỗng, sao cho $L = N(A) = T(B)$.

Chứng minh: Vấn đề chính là xây dựng PDA B để thoả mãn điều kiện trên.

B được xây dựng như sau:

$B = (Q \cup \{q_0', d\}, \Sigma, \Gamma \cup \{Z_0'\}, \delta', q_0', Z_0', \varnothing)$, trong đó

R1: $\delta'(q_0', \Lambda, Z_0') = \{(q_0, Z_0Z_0')\}$

R2: $\delta'(q, a, Z) = \delta(q, a, Z)$ với mọi $a \in \Sigma$, $q \in Q$, $Z \in \Gamma$

R3: $\delta'(q, \Lambda, Z) = \delta(q, \Lambda, Z) \cup \{(d, \Lambda)\}$ với mọi $Z \in \Gamma \cup \{Z_0'\}$, $q \in F$

R4: $\delta'(d, \Lambda, Z) = \{(d, \Lambda)\}$, với mọi $Z \in \Gamma \cup \{Z_0'\}$

Chúng ta có thể khẳng định $L = T(A) = N(B)$. ■

Ví dụ 6.5 Xây dựng PDA để đoán nhận được tập các xâu trên {a, b} trong đó số chữ a bằng số các chữ b.

Giải: $B = (\{q\}, \{a, b\}, \{Z_0, a, b\}, \delta', q, Z_0, \varnothing)$, trong đó

$\delta(q, a, Z_0) = \{(q, aZ_0)\}$, $\quad$ $\delta(q, b, Z_0) = \{(q, bZ_0)\}$

$\delta(q, a, a) = \{(q, aa)\}$, $\quad$ $\delta(q, b, b) = \{(q, bb)\}$

$\delta(q, a, b) = \{(q, \Lambda)\}$, $\quad$ $\delta(q, b, a) = \{(q, \Lambda)\}$

$\delta(q, \Lambda, Z_0) = \{(q, \Lambda)\}$

Chứng minh được rằng $N(A) = L$ là tập tất cả các xâu trên {a, b} trong đó số các chữ a bằng số các chữ b.

6.3 Ôtômát đẩy xuống PDA và ngôn ngữ phi ngữ cảnh

Định lý 6.3 Nếu L là ngôn ngữ phi ngữ cảnh thì sẽ tồn tại PDA A để đoán nhận L bằng kho đẩy xuống rỗng, $N(A) = L$.

Chứng minh: Dựa vào văn phạm phi ngữ cảnh G sinh ra L để xây dựng PDA A.

Bước 1. Xây dựng PDA M. L = L(G), trong đó G = (V_N, Σ, P, S) là văn phạm phi ngữ cảnh. PDA được xây dựng như sau:

A = ({q}, Σ, $V_N \cup \Sigma$, δ, q, S, ∅), trong đó δ được định nghĩa như sau:

R_1: $\delta(q, \Lambda, A) = \{(q, \alpha) \mid A \to \alpha$ thuộc P}

R_2: $\delta(q, a, a) = \{(q, \Lambda)\}$ với mọi $a \in \Sigma$.

Nếu $w \in L(G)$ thì nó được dẫn xuất trái nhất ra từ S

$$S \to u_1A_1\alpha_1 \Rightarrow u_1u_2A_2\alpha_2\alpha_1 \Rightarrow \ldots \Rightarrow w,$$

nên A có thể có kho đẩy xuống là rỗng. Trước tiên A thực hiện Λ- chuyển dịchtương ứng với $S \to u_1A_1\alpha_1$. Ôtômát A sẽ xoá S và lưu vào kho $u_1A_1\alpha_1$. Sau đó A lại sử dụng các qui tắc R_2 để xoá đi những ký hiệu (kết thúc) trong u_1 và ký hiệu còn lại trên đỉnh của kho đẩy xuống là A_1 (ký hiệu không kết thúc). Tiếp tục sử dụng dẫn xuất $A_1 \to u_2A_2\alpha_2$ và thực hiện tương tự như trên cho đến khi đi A kho đẩy xuống là rỗng khi đã đọc xong toàn bộ w.

Bước 2. Phần còn lại là chứng minh L(G) = N(A) bằng phương pháp chứng minh qui nạp theo các qui tắc kiến thiết.

Trước tiên ta chứng minh $L(G) \subseteq N(A)$. Giả sử $w \in L(G)$, suy ra nó có thể là kết quả của một dẫn xuất trái nhất. Trong mỗi bước của dẫn xuất trái nhất luôn có dạng: $uA\propto$, với $u \in \Sigma^*$, $A \in V_N$, $\propto \in (V_N \cup \Sigma)^*$. Ta cần chứng minh rằng nếu có dẫn xuất $S \overset{*}{\Rightarrow} uA\propto$ thì

$$(q, uv, Z_0) \vdash^* (q_0, v, A\propto), \forall v \in \Sigma^* \qquad (6.11)$$

Ta chứng minh tính chất trên qui nạp theo các bước trong dẫn xuất ra $uA\propto$. Nếu $S \overset{0}{\Rightarrow} uA$, thì $u = \propto = \Lambda$ và A = S. Hiển nhiên, cơ sở qui nạp là đúng. Giả thiết (6.11) đúng với dẫn xuất qua n bước.

Ta xét $S \overset{n+1}{\Rightarrow} uA\propto$ là một dẫn xuất trái nhất. Tất nhiên ta có thể viết thành

$S \overset{n}{\Rightarrow} u_1A_1\propto_1 \Rightarrow uA\propto$. Ở bước cuối ta áp dụng A_1-dẫn xuất: $A_1 \to u_2A\propto_2$, với $u = u_1u_2$, $\propto = \propto_1\propto_2$.

Áp dụng giả thiết qui nạp với n bước đầu, ta có

$$(q, u_1u_2v, S) \vdash^* (q, u_2v, A\propto_1) \qquad (6.12)$$

Bởi vì $A_1 \to u_2A\propto_2$ là một qui tắc trong G, nên theo R_1 ta xây có

$(q, \Lambda, A_1) \vdash (q, \Lambda, u_2A\propto_2)$. Áp dụng các tính chất của hàm chuyển trạng thái,

$$(q, u_2v, A_1\propto_1) \vdash (q, u_2v, u_2A\propto_2\propto_1)$$

$$\vdash (q, v, A\propto_2\propto_1)$$

Nghĩa là, $(q, u_2v, S) \vdash^* (q, v, A\propto_2\propto_1)$ (6.13)

Do $u = u_1u_2$, $\propto = \propto_1\propto_2$, nên kết hợp (6.12) với (6.13), ta được

$$(q, uv, S) \vdash^* (q, v, A\propto)$$

Tiếp theo ta chứng minh $L(G) \subseteq N(A)$. Giả sử $w \in L(G)$, suy ra có một dẫn xuất trái nhất

$$S \overset{*}{\Rightarrow} uAv \Rightarrow uu'v = w.$$

Từ (6.12) suy ra

$$(q, uu'v, S) \vdash^* (q, u'v, Av)$$

Vì $A \to u'$ là qui tắc trong P, nên $(q, u'v, A) \vdash^* (q, u'v, u'v)$

Theo qui tắc R_2, ta có $(q, u'v, u'v) \vdash^* (q, \Lambda, \Lambda)$, nghĩa là $w = uu'v \in N(A)$ khẳng định $L(G) \subseteq N(A)$.

Tiếp theo ta chứng minh chiều ngược lại $L(G) \subseteq N(A)$.

Tương tự, ta chứng minh một tính chất bổ trợ

Nếu $(q, uv, S) \vdash^* (q, v, \propto)$ thì $S \overset{*}{\Rightarrow} u\propto$ (6.14)

Chứng minh (6.14) qui nạp theo các bước dịch chuyển trạng thái của PDA.

Bước cơ sở qui lạp là hiển nhiên đúng, vì nếu $(q, uv, S) \vdash^0 (q, v, \propto)$ thì $u = \Lambda$, $S = \propto$, thì tất nhiên $S \overset{0}{\Rightarrow} \Lambda \propto$.

Giả sử (6.14) đúng với n bước dịch chuyển. Ta xét trường hợp với n + 1 bước dịch chuyển trạng thái

$$(q, uv, S) \vdash^{n+1} (q, v, \propto) \quad (6.15)$$

Trong bước dịch chuyển trạng thái cuối cùng trong hình trạng (6.15) của PDA có thể nhận được từ một trong hai dạng sau: $(q, \Lambda, A) \vdash (q, \Lambda, \propto')$ hoặc $(q, a, a) \vdash (q, \Lambda, \Lambda)$. Theo trường hợp đầu thì (6.15) có thể viết thành

$$(q, uv, S) \vdash^n (q, v, A\propto_2) \vdash (q, v, \propto_1\propto_2) = (q, v, \propto)$$

Theo giả thiết qui nạp thì $S \overset{*}{\Rightarrow} uA\propto_2 \Rightarrow u\propto_1\propto_2 = u\propto$

Đối với trường hợp thứ hai ta có thể chia ra

$$(q, uv, S) \vdash^n (q, av, a\propto) \vdash (q, v, \propto)$$

Ta có thể viết $u = u'a$, với $u' \in \Sigma$. Do vậy $(q, u'a\, v, S) \vdash^n (q, av, a\propto)$ suy ra (theo giả thiết qui nạp) $S \overset{*}{\Rightarrow} u'a\propto = u\propto$.

Kết hợp cả hai trường hợp, ta đã chứng minh được tính chất (6.14).

Phần còn lại ta phải chứng minh nếu $w \in N(A)$ thì $w \in L(G)$.

Thật vậy, $w \in N(A)$, tức là $(q, w, S) \vdash^* (q, \Lambda, \Lambda)$. Đặt $u = w$, $v = \propto = \Lambda$ và áp dụng (6.14), ta nhận được $S \overset{*}{\Rightarrow} w\Lambda = w$, nghĩa là $w \in L(G)$.

Cuối cùng ta đã chứng minh được $L(G) = N(A)$ ■

Ví dụ 6.6 Xây dựng ôtômát đẩy xuống A tương đương với văn phạm phi ngữ cảnh có các qui tắc: $S \to 0BB$, $B \to 0S \mid 1S \mid 0$. Kiểm tra xem $010^4 \in N(M)$?

Giải: Định nghĩa PDA A như sau:

$A = (\{q\}, \{0, 1\}, \{S, B, 0, 1\}, \delta, q, S, \varnothing)$, trong δ được định nghĩa như sau:

R_1: $\delta(q, \Lambda, S) = \{(q, 0BB)\}$

R_2: $\delta(q, \Lambda, B) = \{(q, 0S), (q, 1S), (q, 0)\}$

R_3: $\delta(q, 0, 0) = \{(q, \Lambda)\}$

R_4: $\delta(q, 1, 1) = \{(q, \Lambda)\}$

Ta kiểm tra xem $010^4 \in N(A)$? Bởi vì

$(q, 010^4) \vdash (q, 010^4, 0BB)$, theo qui tắc R_1

$\vdash (q, 10^4, BB)$, theo qui tắc R_3

$\vdash (q, 10^4, 1SB)$, theo qui tắc R_2 bởi vì $(q, 1S) \in \delta(q, \Lambda, B)$

$\vdash (q, 0^4, SB)$, theo qui tắc R_4

$\vdash (q, 0^4, 0BBB)$, theo qui tắc R_1

$\vdash (q, 0^3, BBB)$, theo qui tắc R_3

$\vdash^* (q, 0^3, 000)$, theo qui tắc R_2 bởi vì $(q, 0) \in \delta(q, \Lambda, B)$

$\vdash^* (q, \Lambda, \Lambda)$, theo qui tắc R_3

Vậy $010^4 \in N(A)$.

Định lý 6.4 Nếu $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ là một PDA thì sẽ tồn tại văn phạm phi ngữ cảnh G sao cho $L(G) = N(A)$.

Chứng minh: Xây dựng văn phạm phi ngữ cảnh G sao cho $L(G) = N(A)$.

Bước 1: Xây dựng $G = (V_N, \Sigma, P, S)$, trong đó

$V_N = \{S\} \cup \{[q, Z, q_1] \mid q, q_1 \in Q, Z \in \Gamma\}$ còn P có các qui tắc:

R_1: S- dẫn xuất: $S \to [q_0, Z_0, q]$ với mọi q trong Q

R_2: Với mọi $(q_1, \Lambda) \in \delta(q, a, Z)$ thì $[q, Z, q_1] \to a \in P$

R_3: Với mọi $(q_1, Z_1Z_2\ldots Z_m) \in \delta(q, a, Z)$ thì

$[q, Z, q'] \to a[q_1, Z_1, q_2][q_2, Z_2, q_3] \ldots [q_m, Z_m, q'] \in P$ với mọi trạng thái $q', q_2, \ldots q_m \in Q$.

Bước 2. Chứng minh L(G) = N(A).

Dựa vào các qui tắc kiến thiết văn phạm để chứng minh tính chất bổ trợ (6.16) qui nạp theo các bước dẫn xuất trong văn phạm và các của chuyển hình trạng trong ôtômát đẩy xuống, tương tự như trong chứng minh định lý 6.3.

$$[q, Z, q'] \overset{*}{\Rightarrow} w \text{ khi và chỉ khi } (q, w, Z) \vdash^* (q', \Lambda, \Lambda) \qquad (6.16)$$

$$w \in L(G) \Leftrightarrow S \overset{*}{\Rightarrow} w$$

$$\Leftrightarrow S \to [q_0, Z_0, q'] \overset{*}{\Rightarrow} w$$

$$\Leftrightarrow (q_0, Z_0, q') \vdash^* (q', \Lambda, \Lambda)$$

$$\Leftrightarrow w \in N(A). \qquad \blacksquare$$

Ví dụ 6.7 Xây dựng văn phạm phi ngữ cảnh G để sinh ra N(M), khi

$M = (\{q_0, q_1\}, \{a, b\}, \{Z_0, Z\}, \delta, q_0, Z_0, \varnothing)$, và δ được xác định như sau:

$\delta(q_0, b, Z_0) = \{(q_0, ZZ_0)\}$

$\delta(q_0, \Lambda, Z_0) = \{(q_0, \Lambda)\}$

$\delta(q_0, b, Z) = \{(q_0, ZZ)\}$

$\delta(q_0, a, Z) = \{(q_1, Z)\}$

$\delta(q_1, b, Z) = \{(q_1, \Lambda)\}$

$\delta(q_1, a, Z_0) = \{(q_0, Z_0)\}$

Giải: Xây dựng $G = (V_N, \{a, b\}, P, S)$, trong đó

$V_N = \{S, [q_0, Z_0, q_0], [q_0, Z_0, q_1], [q_0, Z, q_0], [q_0, Z, q_1], [q_1, Z_0, q_0], [q_1, Z_0, q_1], [q_1, Z, q_0], [q_1, Z, q_1]\}$

P bao gồm những qui tắc sau:

P_1: $S \to [q_0, Z_0, q_0]$

P_2: $S \to [q_0, Z_0, q_1]$

Bởi vì $\delta(q_0, b, Z_0) = \{(q_0, ZZ_0)\}$ nên

P_3: $[q_0, Z_0, q_0] \to b[q_0, Z, q_0][q_0, Z_0, q_0]$

P_4: $[q_0, Z_0, q_0] \to b[q_0, Z, q_1][q_1, Z_0, q_0]$

P_5: $[q_0, Z_0, q_1] \to b[q_0, Z, q_0][q_0, Z_0, q_1]$

P_6: $[q_0, Z_0, q_1] \to b[q_0, Z, q_1][q_1, Z_0, q_1]$

Vì $\delta(\backslash q_0, b, Z_0) = \{(q_0, \Lambda)\}$ nên

P_7: $[q_0, Z_0, q_0] \to \Lambda$

Vì $\delta(q_0, b, Z) = \{(q_0, ZZ)\}$ nên

P_8: $[q_0, Z, q_0] \to b[q_0, Z, q_0][q_0, Z, q_0]$

P_9: $[q_0, Z, q_0] \to b[q_0, Z, q_1][q_1, Z, q_0]$

P_{10}: $[q_0, Z, q_1] \to b[q_0, Z, q_0][q_0, Z, q_1]$

P_{11}: $[q_0, Z, q_1] \to b[q_0, Z, q_1][q_1, Z, q_1]$

$\delta(q_0, a, Z) = \{(q_1, Z)\}$ nên ta có

P_{12}: $[q_0, Z, q_0] \to a[q_1, Z, q_0]$

P_{13}: $[q_0, Z, q_1] \to a[q_1, Z, q_1]$

$\delta(q_1, b, Z_0) = \{(q_0, \Lambda)\}$ nên ta có

P_{14}: $[q_1, Z, q_1] \to b$

$\delta(q_1, a, Z_0) = \{(q_0, Z_0)\}$ nên ta có

P_{15}: $[q_1, Z_0, q_0] \to a[q_0, Z_0, q_0]$

P_{16}: $[q_1, Z_0, q_1] \to a[q_0, Z_0, q_1]$

Lưu ý: Có thể sử dụng kỹ thuật như ở chương 5 để lược bỏ những biến, qui tắc dư thừa trong văn phạm G.

6.4 Phân tích cú pháp và ôtômát đẩy xuống

Ôtômát đẩy xuống PDA được sử dụng nhiều trong phân tích cú pháp của các ngôn ngữ tự nhiên và các ngôn ngữ lập trình. Có hai cách phân tích cú pháp: phân tích trên / xuống và dưới / lên.

6.4.1 Phân tích cú pháp trên / xuống

Trong phần này chúng ta nghiên cứu kỹ thuật phân tích cú pháp trên / xuống được ứng dụng trong một số lớp con của ngôn ngữ phi ngữ cảnh.

Ví dụ 6.8 Cho G = ({S, A, B}, {a, b}, P, S), trong đó P có $S \to aAB$, $S \to bBA$, $A \to bS$, $A \to a$, $B \to aS$, $B \to b$ và $w = abbbab \in L(G)$.

Suy diễn bên trái nhất để sinh ra w:

$$S \Rightarrow aAB \Rightarrow abSB \Rightarrow abbBAB \Rightarrow abbbAB \Rightarrow abbbaB \Rightarrow abbbab$$

Để thực hiện được các dẫn xuất như trên thì phải xác định được những qui tắc nào sẽ được áp dụng lần lượt. Một trong các kỹ thuật được sử dụng trong phân tích cú pháp trên / xuống là thành lập bảng từ vựng.

Để tiện lợi chúng ta ký hiệu các qui tắc áp dụng để có suy dẫn trên: $S \to aAB$, $A \to bS$, $S \to bBA$, $A \to a$, $B \to aS$, $B \to b$ tương ứng là $P_1, P_2, \ldots, P_6$. Ký hiệu E cho lỗi khi xâu vào không nằm trong L(G).

Bảng 6.1 Bảng phân tích từ vựng

	Λ	a	b
S	E	P_1	P_2
A	E	P_4	P_3
B	E	P_5	P_6

Nếu A là biến bên trái nhất trong dẫn xuất và biến đầu tiên trong dãy con các ký hiệu vào chưa được xử lý là *b* thì sử dụng qui tắc P_3.

Văn phạm có tính chất trên, chỉ cần tìm về phía trước một ký hiệu vào thì đã quyết định được qui tắc nào cần áp dụng tiếp theo, được gọi là *văn phạm LL(1).*

Ví dụ 6.9 Cho trước văn phạm phi ngữ cảnh G có các qui tắc: $S \rightarrow F + S \mid F * S \mid F$, $F \rightarrow a$. Xác định dãy phân tích cú pháp trên / xuống cho xâu w = a + a * a.

Nếu chỉ tìm về phía trước một ký hiệu thì chúng ta sẽ không phân tích được w. Đối với xâu w, có thể áp dụng $F \rightarrow a$ để sinh ra a. Nhưng nếu a đứng sau dấu + hoặc * thì không thể áp dụng để sinh ra a được. Nghĩa là chúng ta phải nhìn về phía trước 2 ký hiệu.

Bắt đầu từ S chúng ta có thể áp dụng ba qui tắc $S \rightarrow F + S \mid F * S \mid F$. Hai ký hiệu đầu của a + a * a là a +. Do vậy chúng ta chỉ có thể áp dụng $S \rightarrow F + S$, sau đó có thể áp dụng $F \rightarrow a$. Vậy $S \Rightarrow F + S \Rightarrow a + S$. Phần còn lại của xâu chưa được phân tích sẽ là a * a. Tương tự hai ký hiệu đầu còn lại là a*. Áp dụng tiếp $S \rightarrow F * S$, $F \rightarrow a$ ta được $S \Rightarrow F + S \Rightarrow a + S \Rightarrow a + F * S \Rightarrow a + a * S$. Ký hiệu còn lại là a, do đó có thể áp dụng $S \rightarrow F$ và $F \rightarrow a$ để phân tích xong cú pháp xâu a + a * a. Vậy, dãy dẫn xuất trái nhất để sinh ra w sẽ là:

$$S \Rightarrow F + S \Rightarrow a + S \Rightarrow a + F * S \Rightarrow a + a * S \Rightarrow a + a * a.$$

Tiếp theo chúng ta xây dựng một bảng để hỗ trợ xác định dẫn xuất trái nhất đối với các xâu đầu vào. Ký hiệu P_1, P_2, P_3, P_4 cho các qui tắc $S \rightarrow F + S$, $S \rightarrow F * S$, $S \rightarrow F$, $F \rightarrow a$ và E ký hiệu lỗi cú pháp.

Bảng 6.2 Bảng phân tích cú pháp cho ví dụ 6.9

	Λ	a	+	*	aa	a+	a*
S	E	P_3	E	E	E	P_1	P_2
F	E	P_4	E	E	E	P_4	P_4
	+a	++	+*	*a	*+	**	
S	E	E	E	E	E	E	
F	E	E	E	E	E	E	

Ví dụ, nếu biến trái nhất là F và hai ký hiệu tiếp theo cần xử lý là a+ thì áp dụng qui tắc P_4. Khi gặp phải hai ký hiệu tiếp theo là *a thì có lỗi E.

Văn phạm có tính chất: phải căn cứ vào *k ký hiệu ở phía trước* mới quyết định được qui tắc nào cần áp dụng trong dẫn xuất tiếp theo, được gọi là *văn phạm LL(k).*

Văn phạm ở ví dụ 6.9 là văn phạm LL(2).

Trong quá trình phân tích cú pháp, vấn đề không tất định gây ra không ít khó khăn. Đó là trường hợp có hai A-dẫn xuất, ví dụ $A \to \alpha\beta$ và $A \to \alpha\gamma$. Vấn đề *nhập nhằng* này được giải quyết bằng kỹ thuật phân tích thành thừ số như sau.

Định lý 6.5 Văn phạm phi ngữ cảnh G có hai A-dẫn xuất: $A \to \alpha\beta$ và $A \to \alpha\gamma$. Nếu thay $A \to \alpha\beta$ và $A \to \alpha\gamma$ bằng $A \to \alpha A'$, $A' \to \beta \mid \gamma$ với A' là biến mới thì sẽ được văn phạm tương đương với G.

Chứng minh: Sự tương đương của hai văn phạm trên dễ dàng được chỉ ra bằng cách thay vì áp dụng $A \to \alpha\beta$ và $A \to \alpha\gamma$, ta sử dụng $A \to \alpha A'$, $A' \to \beta \mid \gamma$ và ngược lại. ■

Một vấn đề khác hay gặp phải trong phân tích cú pháp là *đệ qui trái.* Biến A được gọi là *đệ qui trái* nếu nó có dạng $A \to A\alpha$. Khi gặp phải trường hợp đệ qui trái thì bộ phân tích cú pháp có thể rơi vào chu trình lặp vô hạn. Định lý sau cho phép loại bỏ đệ qui trái.

Định lý 6.6 Cho G là văn phạm phi ngữ cảnh có tập tất cả các A-dẫn xuất là:

$\{ A \to A\alpha_1, \ldots, A \to A\alpha_n, A \to \beta_1, \ldots, A \to \beta_m\}$. Văn phạm G_1 được xây dựng từ G bằng cách thêm biến mới A' và thay các A-dẫn xuất trong G bằng $A \to \beta_1 A'$, …, $A \to \beta_m A'$, $A' \to \alpha_1 A'$, …, $A' \to \alpha_n A'$ và $A' \to \Lambda$ là tương đương với G.

Chứng minh: Tương tự như đối với bổ đề 5.3.

Định lý 6.5 và 6.6 chỉ áp dụng dụng hiệu quả để xây dựng bộ phân tích cú pháp top/down cho một số văn phạm phi ngữ cảnh, không phải cho tất cả cc trường hợp. Chúng ta khái quát quá trình xây dựng bộ phân tích cú pháp top/down như sau:

Xây dựng bộ phân tích cú pháp top/down (trên / xuống)

1. Loại đệ qui trái bằng cách sử dụng định lý 6.6 cho tất cả các biến đệ qui trái.
2. Áp dụng định lý 6.5 để loại bỏ sự nhập nhằng khi cần thiết.
3. Nếu văn phạm thu được là LL(k) với k là số tự nhiên thì áp dụng kỹ thuật như đã nêu ở hai ví dụ trên để phân tích qua k bước trên/xuống.

Ví dụ 6.10 Xét ngôn ngữ gồm tất cả các biểu thức số học với phép +, * trên các biến x1 và x2. Văn phạm sinh ra L có dạng:

G = ({T, F, E}, {x, 1, 2, +, *, (,)}, P, E), trong đó P có các qui tắc:

$E \to E + T$	$F \to (E)$	
$E \to T$	$F \to x1$	
$T \to T * F$	$F \to x2$	$T \to F$

Xây dựng bộ phân tích cú pháp cho L(G).

Giải:

Bước 1: Áp dụng định lý 6.6 để loại bỏ đệ qui trái: E và T. Thay $E \to E + T$ và $E \to T$ bằng $E \to TE'$, $E' \to + TE'$ và $E' \to \Lambda$, E′ là biến mới bổ sung. Tương tự thay $T \to T * F$ và $T \to F$ bằng $T \to FT'$, $T' \to + FT'$ và $T' \to \Lambda$. Văn phạm mới G_1 tương với G là:

$G_1 = (\{T, F, E, E', T'\}, \{x, 1, 2, +, *, (,)\}, P', E)$, P′ có các qui tắc:

$E \to TE'$	$F \to (E)$	$T \to FT'$	$T' \to \Lambda$
$E' \to + TE'$	$E' \to \Lambda$	$F \to x1$	$F \to x2$
$T' \to * FT'$			

Bước 2: Áp dụng định lý 6.5 để loại bỏ sự nhập nhằng bằng kỹ thuật phân tích thành thừa số. Thay $F \to x1$, $F \to x2$ bằng $F \to xN$, $N \to 1 \mid 2$. Kết quả được văn phạm G_2 tương đương:

$G_2 = (\{T, F, E, E', T'\}, \{x, 1, 2, +, *, (,)\}, P'', E)$, P″ có các qui tắc:

P_1: $E \to TE'$	P_6: $T' \to \Lambda$
P_2: $E' \to + TE'$	P_7: $F \to (E)$
P_3: $E' \to \Lambda$	P_8: $F \to xN$
P_4: $T \to FT'$	P_9: $N \to 1$
P_5: $T' \to * FT'$	P_{10}: $N \to 2$

Bước 3: Xây dựng bảng phân tích từ vựng cho văn phạm LL(1) (ở bước 2)

Bảng 6.3 Bảng phân tích từ vựng

	Λ	x	1	2	+	*	(	)
E	E	P_1	E	E	E	E	P_1	E
T	E	P_4	E	E	E	E	P_4	E
F	E	P_8	E	E	E	E	P_1	E
T′	P_6	E	E	E	E	P_5	E	P_6
E′	P_1	E	E	E	P_2	E	E	P_1
N	E	E	P_9	P_{10}	E	E	E	E

6.4.2 Phân tích cú pháp dưới / lên

Phân tích từ dưới lên trên dựa chủ yếu vào cây dẫn xuất đối với dãy đầu vào cho trước. Đối với lớp các văn phạm *quyền ưu tiên yếu* chúng ta có thể xây dựng PDA đơn định như là bộ phân tích cú pháp để phân tích cú pháp cho những xâu đoán nhận bởi văn phạm đó.

Trong phân tích cú pháp dưới / lên ta cần phải đảo ngược các qui tắc để cuối cùng nhận được S. PDA hoạt động như bộ phân tích dưới / lên:

(i) $\delta(q, \Lambda, \alpha^T) = \{(q, \Lambda) \mid \text{có qui tắc } A \to \alpha\}$

(ii) $\delta(q, \sigma, \Lambda) = \{(q, \sigma) \mid$ với mọi σ trong $\Sigma\}$

Áp dụng (i) để thay α^T bằng A khi có qui tắc $A \to \alpha$. Ký hiệu vào σ được lấy ra khỏi kho bằng cách sử dụng (ii). Để được đoán nhận ta cần một số dịch chuyển khi PDS có S hoặc Z_0 ở đỉnh của kho đẩy xuống.

Trong phân tích cú pháp dưới / lên, chúng ta cần xây dựng PDA để đoán nhận L(G)$. Ở đây chúng ta có hai thao tác:

(i) Chuyển dịch ký hiệu vào lên đỉnh của kho Stack.
(ii) Thay thế α^T bằng A khi có qui tắc $A \to \alpha$ trong G.

Ở mỗi bước chúng ta phải quyết định xem thực hiện thao tác nào. Để lựa chuyển vào Stack (i) hay thay thế (ii), chúng ta sử dụng quan hệ R được gọi là *quan hệ thứ tự ưu tiên*.

Nếu $(a, b) \in R$, trong đó a là ký hiệu ở đỉnh của kho Stack, b là ký hiệu vào thì thực hiện thay thế (ii), ngược lại thực hiện chuyển vào Stack (i).

Ví dụ 6.11 Xây dựng bộ phân tích cú pháp dưới / lên cho ngôn ngữ L(G)$, trong đó G được cho ở ví dụ 6.10.

Giải:

G có các qui tắc $E \to E + T$, $F \to (E)$, $E \to T$, $F \to x1$, $T \to T * F$, $F \to x2$, $T \to F$.

Trước tiên ta xây dựng quan hệ thứ tự ưu tiên R. Trong bảng 6.4, ta ký hiệu a có quan hệ R với b là '√' ở ô tương ứng.

Bảng 6.4 Quan hệ thứ tự ưu tiên

Ký hiệu ở kho/ Ký hiệu vào	x	(	)	1	2	+	*	$
Z_0								
x								
(								
)			√			√	√	√
1			√			√	√	√
2			√			√	√	√
+								
*								
E								
T			√			√		√
F			√			√	√	√

Sử dụng bảng 6.4 để thực hiện các thao tác cần thiết. Ví dụ, nếu ký hiệu ở Stack là F và ký hiệu vào tiếp theo là * thì thực hiện thay thế thay thế (ii). Nếu ký hiệu ở kho là E thì đặt các ký hiệu vào đỉnh của kho Stack.

Ôtômát đẩy xuống đơn định hoạt động như một bộ phân tích cú pháp được xây dựng như sau:

$A = (Q, \Sigma', \Gamma, \delta, p, Z_0, \varnothing)$, trong đó

$\Sigma' = \Sigma \cup \{\$\}$, $Q = \{p\} \cup \{p_\sigma \mid \sigma \in \Sigma'\}$

$\Gamma = \{E, T, F\} \cup \Sigma \cup \{Z_0, \$\}$

δ được xác định như sau:

R_1: $\delta(p, \sigma, \Lambda) = (p, \Lambda)$

R_2: $\delta(p_\sigma, \Lambda, \alpha) = (p, \sigma\alpha)$, với mọi $\sigma, \alpha \notin R$

R_3: $\delta(p_\sigma, \Lambda, T + E) = (p_\sigma, E)$, khi $(T, \sigma) \in R$

R_4: $\delta(p_\sigma, \Lambda, T\alpha) = (p_\sigma, E\alpha)$, khi $(T, \sigma) \in R$ và $\alpha \in \Gamma - \{+\}$

R_5: $\delta(p_\sigma, \Lambda, F * T) = (p_\sigma, T)$, khi $(F, \sigma) \in R$

R_6: $\delta(p_\sigma, \Lambda, F\alpha) = (p_\sigma, T\alpha)$, khi $(F, \sigma) \in R$ và $\alpha \in \Gamma - \{*\}$

R_7: $\delta(p_\sigma, \Lambda, EC) = (p_\sigma, F)$, khi $(), \sigma) \in R$

R_8: $\delta(p_\sigma, \Lambda, 1x) = (p_\sigma, F)$, khi $(1, \sigma) \in R$

R_9: $\delta(p_\sigma, \Lambda, 2x) = (p_\sigma, F)$, khi $(2, \sigma) \in R$

R_{10}: $\delta(p_\$, \Lambda, E) = (p_\$, \Lambda)$

R_{11}: $\delta(p_\$, \Lambda, Z_0) = (p_\$, \Lambda)$

Bộ phân tích cú pháp dưới / lên cho xâu vào x1 + x2 được cho như ở bảng 6.5.

Bảng 6.5 Phân tích cú pháp của x1 + (x2)

Bước	Trạng thái	Dãy vào Chưa được đọc	Kho Stack	Luật sử dụng	Qui tắc áp dụng
1	p	x1 + (x2)	Z_0	-	
2	p_x	1 + (x2)	Z_0	R_1	
3	p	1 + (x2)	xZ_0	R_2	
4	p_1	+ (x2)	xZ_0	R_1	
5	p	+ (x2)	$1xZ_0$	R_2	
6	p_+	(x2)	$1xZ_0$	R_1	
7	p_+	(x2)	FZ_0	R_8	$F \to x1$
8	p_+	(x2)	TZ_0	R_6	$T \to F$
9	p_+	(x2)	EZ_0	R_4	$E \to T$
10	p	(x2)	$+EZ_0$	R_2	
11	$p_($	x2)	$+EZ_0$	R_1	
12	p	x2)	$(+EZ_0$	R_2	
13	p_x	2)	$(+EZ_0$	R_1	
14	p	2)	$x(+EZ_0$	R_2	
15	p_2	)	$x(+EZ_0$	R_1	
16	p	)	$2x(+EZ_0$	R_2	
17	$p_)$	$	$2x(+EZ_0$	R_1	
18	$p_)$	$	$F(+EZ_0$	R_9	$F \to x1$

19	$p_{)}$	$	$T(+EZ_0$	R_6	$T \to F$
20	$p_{)}$	$	$E(+EZ_0$	R_4	$E \to T$
21	p	$	$)E(+EZ_0$	R_2	
22	$p_{\$}$	Λ	$)E(+EZ_0$	R_1	
23	$p_{\$}$	Λ	$F+EZ_0$	R_7	$F \to (E)$
24	$p_{\$}$	Λ	$T+EZ_0$	R_6	$T \to F$
25	$p_{\$}$	Λ	EZ_0	R_3	$F \to E+T$
26	$p_{\$}$	Λ	Z_0	R_{10}	
27	$p_{\$}$	Λ	Λ	R_{11}	

Suy ngược lại các qui tắc đã được áp dụng, chúng ta có dãy các dẫn xuất phải nhất để đoán nhận x1 + (x2) như sau:

$$E \Rightarrow E + T \Rightarrow E + F \Rightarrow E + (E) \Rightarrow E +(T) \Rightarrow E + (F) \Rightarrow E + (x2) \Rightarrow T + (x2)$$
$$\Rightarrow F + (x2) \Rightarrow x1 + (x2).$$

Bài tập về ôtômát đẩy xuống

6.1 Xây dựng PDA để đoán nhận bằng kho đẩy xuống rỗng các ngôn ngữ sau:

(a) $\{a^n b^m c^n \mid m, n \geq 1\}$

(b) $\{a^n b^{2n} \mid n \geq 1\}$

(c) $\{a^m b^m c^n \mid m, n \geq 1\}$

(d) $\{a^m b^n \mid m > n \geq 1\}$

6.2 Xây dựng PDA để đoán nhận bằng trạng thái kết thúc các ngôn ngữ như trong bài 6.1

6.3 Xây dựng văn phạm phi ngữ cảnh sinh ra các ngôn ngữ sau và sau đó xây dựng các PDA để đoán nhận chúng bằng kho đẩy xuống rỗng:

(a) $\{a^n b^n \mid n \geq 1\} \cup \{a^m b^{2m} \mid m \geq 1\}$

(b) $\{a^n b^m c^n \mid m, n \geq 1\} \cup \{a^n c^n \mid n \geq 1\}$

(c) $\{a^n b^m c^m d^n \mid m, n \geq 1\}$

6.4 Xây dựng ôtômát đẩy xuống đơn định trên {a, b} để đoán nhận tất cả các dãy Palindrrome có độ dài là chẵn theo kho đẩy xuống (ngăn xếp) rỗng.

6.5 Chỉ ra rằng tập tất cả các xâu trên {a, b} có số ký tự a bằng số ký tự b, được đoán nhận bởi một ôtômát đẩy xuống đơn định.

6.6 Chỉ ra rằng $\{a^n b^n \mid n \geq 1\} \cup \{a^m b^{2m} \mid m \geq 1\}$ không thể đoán nhận bởi một ôtômát đẩy xuống đơn định.

6.7 Chỉ ra rằng mọi tập chính qui được đoán nhận bởi ôtômát hữu hạn với n trạng thái sẽ được đoán nhận bởi ôtômát đẩy xuống đơn định với một trạng thái và n ký hiệu đẩy xuống.

6.8* (Đề thi cao học năm 2001, câu 3 môn thi cơ bản: "Toán học rời rạc")

Cho ngôn ngữ phi ngữ cảnh $L = \{a^n cb^m cd^k \mid n > m > 0,\ k > 0\}$

(i) Tìm văn phạm phi ngữ cảnh G để sinh ra L (L(G) = L).

(ii) Chỉ ra một dẫn xuất chứng tỏ từ $a^3cb^2cd^2$ được sinh ra bởi G mà anh (chị) đã chỉ ra ở trên.

(iii) Xây dựng cây cú pháp của từ $a^3cb^2cd^2$.

(iv) Xây dựng ôtômát đẩy xuống M để đoán nhận L cho trước ở trên.

(v) Chỉ ra một dẫn xuất chứng tỏ từ $a^3cb^2cd^2$ được đoán nhận bởi M mà anh (chị) đã chỉ ra ở trên.

6.9* (Đề thi cao học năm 2003, câu 3 môn thi cơ bản: "Toán học rời rạc")

Cho văn phạm phi ngữ cảnh G có tập các qui tắc $P = \{S \rightarrow aSa \mid bSb \mid cSc \mid d\}$

(i) Tìm ngôn ngữ sinh L = L(G).

(ii) Xây dựng ôtômát đẩy xuống M để đoán nhận L cho trước ở trên theo ngăn xếp rỗng, L(G) = L(M).

(iii) Cho w_1 = aababbccdccbbabaa và w_2 = ababbcccdbbabaa. Chỉ ra rằng $w_1 \in L(G)$ và $w_1 \in L(M)$, còn $w_1 \notin L(G)$ và $w_1 \notin L(M)$.

(iv) Xây dựng cây cú pháp của xâu w_1 đối với văn phạm G nêu trên.

CHƯƠNG VII
Văn phạm LR(k)

Chương VII giới thiệu về văn phạm LR(k), một tập con của lớp các văn phạm phi ngữ cảnh. Văn phạm LR(k) đóng vai trò quan trọng trong việc nghiên cứu các tính chất của ngôn ngữ lập trình và thiết kế chương trình dịch ([3], [4], [5]). Ví dụ, ngôn ngữ lập trình truyền thống họ ALGOL có bộ phân tích cú pháp tương ứng với văn phạm LR(1).

7.1 Văn phạm LR(k)

Trong các chương trước, chúng ta đã đề cập đến cách sử dụng các qui tắc dẫn xuất của một văn phạm để sinh ra các xâu, các câu của ngôn ngữ và cách kiểm tra xem một từ có thuộc ngôn ngữ sinh bởi văn phạm đó hay không. Trong thiết kế các ngôn ngữ lập trình và thiết kế chương trình dịch, vấn đề rất quan trọng là phải phát triển kỹ thuật phân tích cú pháp, kỹ thuật tìm "dẫn xuất ngược" trong ngôn ngữ phi ngữ cảnh, nghĩa là tìm cây dẫn xuất của các câu *w* cho trước trong ngôn ngữ phi ngữ cảnh. Hầu hết các ngôn ngữ lập trình bậc cao như Pascal, ngôn ngữ C đều là ngôn ngữ phi ngữ cảnh. Quá trình phân tích để xác định xem một biểu thức, một câu lệnh có chính xác về mặt văn phạm hay không, nghĩa là nó có phải một câu suy diễn hay không, là công việc rất quan trọng của các chương trình dịch, thông dịch.

Để xác định được cây dẫn xuất cho một câu (xâu) *w* cho trước, chúng ta phải bắt đầu phân tích *w* và thay những dãy con w_1 của *w* bằng biến *A*, nếu có $A \rightarrow w_1$ là một qui tắc dẫn xuất. Chúng ta lặp lại quá trình thay thế này cho đến khi nhận được biến bắt đầu *S*. Thông thường trong mỗi lần tìm các xâu con để thay thế, chúng ta có một số khả năng lựa chọn và chỉ chọn một trong số đó. Nếu chúng ta chọn một qui tắc dẫn xuất và sau đó một số bước có thể không chọn được qui tắc nào để thay thế tiếp thì không nhận được ký hiệu bắt đầu *S*. Trong những trường hợp như thế, chúng ta lại phải quay lại lần thế trước đó và thử tiếp với những xâu con khác. Trong mọi trường hợp, việc quay lui mà không nhận được ký hiệu bắt đầu thì có lỗi cú pháp, nghĩa là w không được sinh ra bởi văn phạm của ngôn ngữ đã cho. Như vậy, quá trình suy diễn ngược của các văn phạm phi ngữ cảnh là không xác định. Một câu hỏi đặt ra là liệu có những lớp con của các văn phạm phi ngữ cảnh mà quá trình thay thế ngược là đơn định hay không? Câu trả lời là có những lớp con các văn phạm phi ngữ cảnh như thế, đó chính là những văn phạm được gọi là LR(k).

Văn phạm LR(k) là loại văn phạm đọc các xâu ký tự vào từ trái qua phải để sinh ra xâu theo dẫn xuất trái nhất bằng cách đọc *k* ký hiệu ở đầu vào.

Trước khi thảo luận chi tiết về văn phạm LR(k), chúng ta phân tích một số cấu trúc cú pháp của xâu ký tự để bước đầu hiểu được ý nghĩa của các câu sinh bởi ngôn ngữ lập trình.

Xét những câu dạng $\alpha\beta w$ của văn phạm phi ngữ cảnh *G*, trong đó $\alpha, \beta \in (V_N \cup \Sigma)^*$ và $w \in \Sigma^*$. Chúng ta quan tâm đến việc tìm một qui tắc dẫn xuất được áp dụng lần cuối để thu được $\alpha\beta w$. Nếu G có qui tắc $A \to \beta$ và nó được áp dụng lần cuối bằng cách duyệt qua *k* ký tự bên phải của β trong $\alpha\beta w$, *văn phạm như thế được gọi là LR(k)*. Qui tắc $A \to \beta$ được gọi là *qui tắc điều khiển* và β là *cán điều khiển*.

Chúng ta viết $\alpha \overset{*}{\underset{R}{\Rightarrow}} \beta$ nếu β được dẫn xuất từ α bằng dẫn xuất phải nhất. Trước khi đưa ra định nghĩa về LR(k) chúng ta xét một văn phạm có thể phân tích cú pháp với một ký hiệu.

Ví dụ 7.1 Giả sử G có các qui tắc $S \to AB$, $A \to aAb$, $A \to \Lambda$, $A \to Bb$, $B \to b$. Dễ nhận thấy $L(G) = \{a^m b^n \mid n > m \geq 1\}$. Một số mẫu câu của G nhận được bằng qui tắc dẫn xuất phải nhất là AB, ABb^k, $a^m A b^m b^k$, $a^m b^{m+k}$, với $k \geq 1$. Bởi vì $S \to AB$. Hiển nhiên AB là cán điều khiển của AB hoặc ABb^k.

+ Nếu áp dụng cho AB thì chúng ta có ngay $S \underset{R}{\Rightarrow} AB$.

+ Nếu áp dụng cán điều khiển với ABb^k thì nhận được $Sb^k \underset{R}{\Rightarrow} ABb^k$. Khi đó không có qui tắc nào để áp dụng tiếp để $S \overset{*}{\underset{R}{\Rightarrow}} ABb^k$.

Do vậy, để quyết định xem AB có phải là cán điều khiển hay không thì phải đọc ký hiệu bên phải của AB. Nếu ký hiệu đó là rỗng Λ thì AB là cán điều khiển. Ngược lại nếu ký hiệu tiếp theo là *b* thì AB không phải là cán điều khiển.

Chúng ta xét tiếp a^2b^3. Duyệt lần lượt các qui tắc từ trái qua phải chúng ta thấy qui tắc điều khiển $A \to \Lambda$ là có thể áp dụng. Khi đó $a^2A\,b^3 \underset{R}{\Rightarrow} a^2\Lambda\,b^3 = a^2\,b^3$, nên có thể chọn $A \to \Lambda$ là qui tắc điều khiển với việc đọc chỉ một ký hiệu (bên phải của Λ).

Định nghĩa 7.1 Giả sử $G = (V_N, \Sigma, P, S)$ là một văn phạm phi ngữ cảnh, trong đó $S \overset{n}{\Rightarrow} S$ chỉ khi $n = 0$. G là văn phạm LR(k), $k \geq 0$ khi

(i) $S \overset{*}{\underset{R}{\Rightarrow}} \alpha A w \underset{R}{\Rightarrow} \alpha\beta w$, với $\alpha, \beta \in V^*_N$ và $w \in \Sigma^*$,

(ii) $S \overset{*}{\underset{R}{\Rightarrow}} \alpha' A' w' \underset{R}{\Rightarrow} \alpha'\beta' w'$, với $\alpha', \beta' \in V^*_N$ và $w' \in \Sigma^*$,

(iii) $|\alpha\beta| + k$ ký hiệu đầu của $\alpha\beta w$ và $\alpha'\beta' w'$ là trùng nhau thì $\alpha = \alpha'$, $A = A'$ và $\beta = \beta'$.

Lưu ý:

1. Nếu $\alpha\beta w$ hoặc $\alpha'\beta' w'$ chứa ít hơn $(|\alpha\beta| + k)$ ký hiệu thì có thể bổ sung thêm các ký hiệu đặc biết như \$ chẳng hạn vào bên phải.
2. Để nhận được cây suy diễn, chúng ta muốn sử dụng những suy diễn "theo thứ tự ngược". Từ $\alpha\beta w$ *và nếu* có qui tắc $A \to \beta$ thì chúng ta phải quyết định xem qui tắc này có được áp dụng ở bước cuối trong suy diễn trái nhất hay không. Xét k ký hiệu ở phía bên kia của β trong $\alpha\beta w$ chúng ta có thể chọn qui tắc $A \to \beta$ ở bước cuối theo yêu cầu. Nếu có $\alpha'\beta' w'$ thoả mãn điều kiện (iii) và có thể áp dụng $A' \to \beta'$ ở bước cuối của suy

diễn phải nhất đối với $\alpha'\beta'w'$. Từ định nghĩa suy ra $A = A'$, $\beta = \beta'$ và $\alpha = \alpha'$. Vậy, $A \to \beta$ chỉ là qui tắc có thể áp dụng và vấn đề này được quyết định trên cơ sở xem xét *k* ký hiệu ở phía bên kia của β. Chúng ta lặp lại quá trình này cho đến khi nhận được S.

3. Nếu G là văn phạm LR(k) thì nó cũng là văn phạm LR(k′) với mọi k′ > k.

Ví dụ 7.2 Cho G là văn phạm có các qui tắc $S \to aA$, $A \to Abb \mid b$. Hãy chỉ ra rằng G là văn phạm LR(0).

Giải: Dễ dàng nhận thấy $L(G) = \{ab^{2n+1} \mid n = 0, 1, 2, ...\}$. Mọi mẫu câu sinh bởi văn phạm này đều là aA, aAb^{2n}, ab^{2n+1}. Giả sử chúng ta cần tìm qui tắc cuối được áp dụng để có được ab^{2n+1}. Trong số vế trái của các qui tắc: aA, Abb, b thì chỉ có qui tắc $A \to b$ là có thể áp dụng được. Việc áp dụng này được quyết định mà không cần xét bất kỳ ký hiệu nào ở bên phải của *b*. Tương tự như vậy đối với hai trường hợp trước đó, có thể quyết định chọn ngay $S \to aA$ hoặc $A \to Abb$ tương ứng để áp dụng. Vậy, G là văn phạm LR(0) .

Ví dụ 7.3 Xét văn phạm G có các qui tắc suy diễn như trong ví dụ 7.1. Hãy chỉ ra rằng G là văn phạm LR(1) nhưng không phải là LR(0) và tìm cây suy diễn cho từ a^2b^4.

Như trong ví dụ 7.1, chúng ta đã khẳng định rằng các câu suy diễn của G được xác định ở bước cuối theo suy diễn phải nhất với một ký hiệu. Do vậy, G là văn phạm LR(1). Tiếp theo chúng ta chỉ ra rằng, $S \to AB$ là qui tắc được áp dụng để suy ra câu AB nhưng không áp dụng để suy ra được ABb^k. Nói cách khác, qui tắc này (là duy nhất) không thể áp dụng mà không cần xét tới ngữ cảnh của ít nhất một ký hiệu. Từ đó suy ra G không phải là LR(0).

Để nhận được cây suy diễn của a^2b^4, chúng ta hãy đọc từ trái qua phải. Khi đọc *a,* chúng ta phải xem xét ký hiệu tiếp theo. Nếu ký hiệu tiếp theo lại là *a* thì đọc tiếp tục, ngược lại, nếu ký hiệu tiếp theo là *b* có thể chọn qui tắc $A \to \Lambda$ làm qui tắc điều khiển và khi đó bước cuối suy diễn phải nhất của a^2b^4 là:

$$a^2Ab^4 \underset{R}{\Rightarrow} a^2\Lambda b^4$$

Tương tự, phân tích a^2Ab^4 từ trái qua phải và nhận thấy aAb có thể sử dụng để suy diễn mà không cần xét tiếp.

$$aAbb^2 \underset{R}{\Rightarrow} a^2Ab^4$$

Tiếp tục sử dụng aAb một lần nữa chúng ta thu được

$$Ab^2 \underset{R}{\Rightarrow} aAbb^2$$

Để tiếp tục suy diễn ngược được, chúng ta lại phân tích Ab^2. Qui tắc $B \to b$ có thể được áp dụng đối với *b* đầu tiên gặp kể từ trái qua phải chứ không phải là *b* cuối.

$$ABb \underset{R}{\Rightarrow} Ab^2$$

Sau đó sử dụng $B \to b$ là qui tắc suy diễn điều khiển

$$AB \underset{R}{\Rightarrow} ABb$$

Cuối cùng áp dụng $S \to AB$

$S \underset{R}{\Rightarrow} AB$

Như vậy, chúng ta có các suy diễn ngược như sau:

$a^2Ab^4 \underset{R}{\Rightarrow} a^2\Lambda b^4$ bằng cách duyệt một ký hiệu tiếp

$aAbb^2 \underset{R}{\Rightarrow} a^2Ab^4$, không cần xét tiếp bất kỳ kỹ hiệu nào cả

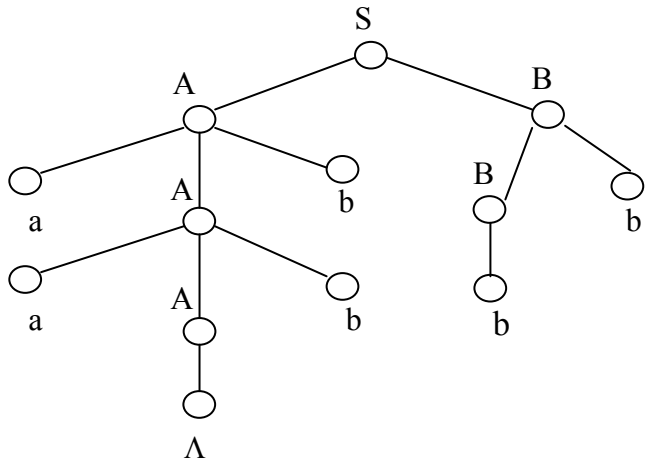
$Ab^2 \underset{R}{\Rightarrow} aAbb^2$, không cần xét tiếp bất kỳ kỹ hiệu nào cả

$ABb \underset{R}{\Rightarrow} Ab^2$, không cần xét tiếp bất kỳ kỹ hiệu nào cả

$AB \underset{R}{\Rightarrow} ABb$, không cần xét tiếp bất kỳ kỹ hiệu nào cả

$S \underset{R}{\Rightarrow} AB$, bằng cách duyệt một ký hiệu tiếp.

và cây suy diễn của từ a^2b^4 có dạng:



Hình H7-1 Cây dẫn xuất để đoán nhận a^2b^4

7.2 Một số tính chất của văn phạm LR(k)

Trong phần này chúng ta giới thiệu một số tính chất quan trọng của các văn phạm LR(k) [4] sử dụng hiệu quả trong phân tích cú pháp và trong nhiều ứng dụng khác.

Trước tiên chúng ta nhắc lại khái niệm nhập nhằng của một văn phạm. Văn phạm G được gọi là nhập nhằng nếu tồn tại từ $w \in L(G)$ mà có hai cây dẫn xuất khác nhau. Tính chất sau khẳng định tính không nhập nhằng của văn phạm LR(k).

Tính chất 7.1 Mọi văn phạm LR(k) đều là văn phạm không nhập nhằng.

Chứng minh: Chúng ta cần chỉ ra rằng với mọi $x \in \Sigma^*$, tồn tại duy nhất một dẫn xuất phải nhất để suy ra *x*. Giả sử có hai dẫn xuất phải nhất cho *x:*

$$S \overset{*}{\underset{R}{\Rightarrow}} \alpha Aw \underset{R}{\Rightarrow} \alpha\beta w = x \qquad (7.1)$$

$$S \overset{*}{\underset{R}{\Rightarrow}} \alpha' A' w' \underset{R}{\Rightarrow} \alpha'\beta' w' = x \qquad (7.2)$$

Bởi vì $\alpha\beta w = \alpha'\beta' w'$, nên từ định nghĩa của văn phạm LR(k) suy ra $\alpha = \alpha'$, $A = A'$ và $\beta = \beta'$. Từ đó suy tiếp, chúng ta có $w = w'$ và $\alpha Aw = \alpha' A' w'$.

Bước cuối trong hai dẫn xuất (7.1) và (7.2) là giống nhau. Lặp lại cách thực hiện như trong (7.1) và (7.2) cho các câu dẫn xuất khác, chúng ta đi đến khẳng định là (7.1) và (7.2) là một. Vì thế, G là không nhập nhằng. ▪

Trong các chương trước chúng ta cũng đã chỉ ra rằng ôtômát đơn định và ôtômát không đơn định có các hành vi như nhau khi xét về khả năng đoán nhận của chúng. Trong chương 6 chúng ta còn chứng minh rằng mọi ôtômát đẩy xuống đều đoán nhận ngôn ngữ phi ngữ cảnh và ngược lại với mọi ngôn ngữ phi ngữ cảnh đều có thể xây dựng ôtômát để đoán nhận ngôn ngữ phi ngữ cảnh đó. Sau đây chúng ta xét một số tính chất xác định mối quan hệ giữa văn phạm LR(k) và ôtômát đẩy xuống.

Tính chất 7.2 Nếu G là văn phạm LR(k), thì sẽ tồn tại ôtômát đẩy xuống đơn định A để đoán nhận L(G).

Tính chất 7.3 Nếu A là ôtômát đẩy xuống đơn định, tồn tại văn phạm LR(1) sao cho L(G) = N(A).

Tính chất 7.4 Nếu G là văn phạm LR(k), với k >1, tồn tại một văn phạm G_1 là LR(1) và tương đương với G.

Định nghĩa 7.2 Ngôn ngữ phi ngữ cảnh được gọi là *đơn định* nếu nó đoán nhận được bởi một ôtômát đẩy xuống đơn định.

Tính chất 7.5 Lớp các ngôn ngữ đơn định là lớp con thực sự của lớp các ngôn ngữ phi ngữ cảnh.

Lớp các ngôn ngữ đơn định ký hiệu là $£_{dc0}$.

Tính chất 7.6 Lớp các ngôn ngữ đơn định đóng với phép lấy phần bù nhưng không đóng với phép hợp và phép giao.

Tính chất 7.7 Một ngôn ngữ phi ngữ cảnh được đoán nhận bởi văn phạm LR(0) khi và chỉ khi nó được đoán nhận bởi ôtômát đẩy xuống và không có một dãy con tiếp đầu ngữ thực sự nào của xâu trong L lại thuộc L.

Tính chất 7.8 Tồn tại thuật toán để kiểm tra xem một văn phạm phi ngữ cảnh có phải LR(k) hay không, với k là số tự nhiên cho trước.

7.3 Các tính chất đóng của các ngôn ngữ

Trong chương 3 và chương 4 chúng ta đã thảo luận một số tính đóng của các ngôn ngữ với các phép hợp, ghép, ... Phần này chúng ta thảo luận về tính đóng của phép giao, phép lấy phần bù của ngôn ngữ. Như đã ký hiệu $£_0$, $£_l$, $£_2$, $£_3$ cho lớp các ngôn ngữ loại *0, cảm ngữ cảnh, phi ngữ cảnh và ngôn ngữ chính qui.*

Tính chất 7.9 Các lớp ngôn ngữ $£_0$, $£_l$, $£_2$, $£_3$ đóng với các phép hợp, phép ghép, phép lấy bao đóng và phép đảo vị (Tính chất 3.5 – 3.7).

Tính chất 7.10 Lớp ngôn ngữ $£_3$ đóng với các phép giao và phép lấy phần bù (Tính chất 4.7 – 4.8).

Tính chất 7.11 Lớp ngôn ngữ $£_2$ không đóng với các phép giao và phép lấy phần bù.

Để khẳng định tính chất 7.11 chúng ta có thể lấy ví dụ minh hoạ.

Như chúng ta đã biết, $L_1 = \{a^n b^n c^i \mid n \geq 1, i \geq 0\}$ và $L_2 = \{a^j b^n c^n \mid n \geq 1, j \geq 0\}$ là hai ngôn ngữ phi ngữ cảnh. Dễ nhận thấy, $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 1\}$ không phải là ngôn ngữ phi ngữ cảnh (xem chương 6). Do vậy, $\pounds_2$ không đóng với phép giao.

Sử dụng luật De Morgan, chúng ta có thể viết $L_1 \cap L_2 = (L_1^c \cup L_2^c)^c$. Tính chất 7.10 đã khẳng định $\pounds_2$ đóng với phép hợp. Nếu $\pounds_2$ đóng với phép lấy phần bù thì L_1^c, L_2^c sẽ là ngôn ngữ phi ngữ cảnh khi L_1, L_2 là các ngôn ngữ phi ngữ cảnh và hợp của chúng cũng sẽ là phi ngữ cảnh. Suy tiếp ra phần bù của hợp các phần bù là phi ngữ cảnh, mâu thuẫn với điều đã khẳng định ở trên. Nghĩa là $\pounds_2$ không đóng với phép lấy phần bù.

Tập các ngôn ngữ chính qui là tập con của các ngôn ngữ phi ngữ cảnh đóng với phép giao, còn lớp các ngôn ngữ phi ngữ cảnh lại không đóng với phép giao. Vậy vấn đề đặt ra là giao của một ngôn ngữ phi ngữ cảnh với ngôn ngữ chính qui sẽ thuộc lớp nào? Tính chất sau sẽ cho ta câu trả lời đó.

Tính chất 7.12 Giao của ngôn ngữ phi ngữ cảnh L với ngôn ngữ chính qui R là một ngôn ngữ phi ngữ.

Chứng minh:

Giả sử L_1 là ngôn ngữ phi ngữ cảnh, L_2 là ngôn ngữ chính qui. Khi đó tồn tại ôtômát đẩy xuống

$M = (Q_1, \Sigma, \Gamma, \delta_1, q_0, Z_0, F_1)$ với $T(M) = L_1$

và ôtômát hữu hạn $A = (Q_2, \Sigma, \delta_2, q_0', F_2)$ để $T(A) = L_2$.

Khi xây dựng một ôtômát đẩy xuống (không đơn định, sau đó chuyển về đơn định), ký hiệu là M^G để đoán nhận $L_1 \cap L_2$ cần phải lưu ý rằng:

+ Mỗi phép chuyển mà M^G thực hiện đều phải tương ứng đồng thời với các phép chuyển của hai ôtômát M và A.

+ Mỗi từ x được đoán nhận bởi M^G thì cũng được đoán nhận đồng thời bởi M và A.

Từ đó xác định được

$M^G = (Q_1 \times Q_2, \Sigma, \Gamma, \delta', [q_0, q_0'], Z_0, F_1 \times F_2)$, trong đó hàm δ' được xác định như sau:

Với $q_1, q_1' \in Q_1$, $q_2, q_2' \in Q_2$, $A \in \Gamma$, $X \in \Gamma^*$ và $a \in \Sigma$

(i) $([q_1', q_2'], X) \in \delta'([q_1, q_2], A, a)$ nếu $(q_1', X) \in \delta_1(q_1, A, a)$ và $q_2' \in \delta_2(q_2, a)$

(ii) $([q_1', q_2'], X) \in \delta'([q_1, q_2], A, \varnothing)$ nếu $(q_1', X) \in \delta_1(q_1, A, \varnothing)$ và $q_2' = q_2$

Qui nạp theo số các phép chuyển có thể chứng minh rằng:

$x \in T(MG)$ khi và chỉ khi $x \in T(M)$ và $x \in T(A)$.

Vậy, $T(M^G) = T(M) \cap T(A)$. ■

Bài tập về văn phạm LR(K)

7.1 Chứng minh rằng văn phạm G có các qui tắc $S \to aAb$, $A \to aAb \mid a$ là LR(1). Nó có là LR(0)?

7.2 Hãy chỉ ra rằng $S \to 0A2$, $A \to 1A1 \mid 1$ không phải là LR(k), $k \geq 0$.

7.3 Cho trước văn phạm $S \to AB \mid a$, $A \to aA \mid aB \mid a$, $B \to b$, tồn tại một số nguyên *k* nào đó để văn phạm này là LR(k)?

7.4 Hãy chỉ ra rằng $\{a^n b^n c^m \mid n, m \geq 1\} \cup \{a^n b^m c^m \mid n, m \geq 1\}$ không phải là LR(k) với mọi *k* nguyên dương.

7.5 Những phát biểu sau có đúng không, tại sao?

(a) Nếu văn phạm G là nhập nhằng, G là LR(k) với k nào đó.

(b) Nếu văn phạm G là không nhập nhằng, G là LR(k) với k nào đó.

7.6 Văn phạm G có các qui tắc: $S \to C \mid D$, $C \to aC \mid b$, $D \to aD \mid C$ có phải là LR(0)?

7.7 Với mỗi qui tắc $A \to \beta$ của văn phạm phi ngữ cảnh G và $w \in \Sigma^*\$^k$ ($\$ \notin V_N \cup \Sigma$), định nghĩa $R_k(w)$ tập tất cả các xâu dạng $\alpha\beta w$ sao cho $A \to \beta$ là điều khiển cho $\alpha\beta ww'$ với $w' \in \Sigma^*\$^*$ và $S\$ \overset{*}{\underset{R}{\Rightarrow}} \alpha Aww' \underset{R}{\Rightarrow} \alpha\beta ww'$. Chứng minh rằng $R_k(w)$ là tập chính qui.

7.8 Chứng minh rằng văn phạm phi ngữ cảnh G là LR(k) nếu và chỉ nếu xâu γ trong $R_k(w)$ tương ứng với qui tắc $A_1 \to \beta_1$ là xâu con của δ trong $R_k(w')$ tương ứng với qui tắc $A_2 \to \beta_2$ suy ra $\gamma = \delta$, $A_1 = A_2$, $\beta_1 = \beta_2$.

CHƯƠNG VIII
Máy Turing, ôtômát giới nội và những bài toán P, NP

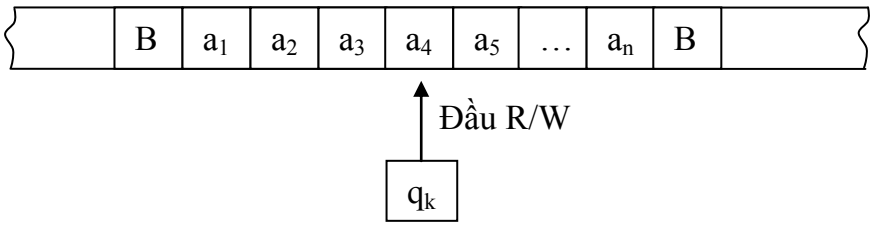
Chương cuối giới thiệu:

- Hoạt động của máy Turing,
- Sự tương đương giữa văn phạm loại 0 và máy Turing,
- Ôtômát tuyến tính giới nội và văn phạm cảm ngữ cảnh,
- Những bài toán quyết định được,
- Độ phức tạp tính toán và những bài toán NP-đầy đủ

8.1 Mô hình máy Turing

Máy Turing là mô hình toán học cho máy tính tổng quát, là nền tảng của quá trình xử lý của máy tính hiện đại, được Alan Turing giới thiệu vào năm 1936. Nói cách khác, máy Turing mô hình được tất cả những khả năng tính toán của máy tính, nghĩa là nó có thể thực hiện được mọi tính toán của bất kỳ máy tính nào. Ngày nay, máy Turing được coi là một mô hình toán học thích hợp nhất để diễn tả khái niệm thuật toán và nhờ đó, các khái niệm về "sự tính được", "giải được" hay tính "quyết định" được xác định một cách hình thức ([2], [4]).

Máy Turing MT gồm một bộ điều khiển với tập hữu hạn thạng thái Q và một đầu đọc/ghi (R/W), chuyển động trên một băng vô hạn cả về hai phía. Băng được chia thành từng ô, mỗi ô chứa một ký hiệu thuộc một bảng ký hiệu hữu hạn Γ, bao gồm cả ký hiệu trắng B (Blank). Máy MT có bảng chữ vào Σ, $\Sigma \subset \Gamma$ và $B \notin \Sigma$. Tại thời điểm bắt đầu hoạt động, dữ liệu vào của MT là dãy ký tự thuộc Σ, được ghi trên các ô liền nhau của băng, các ô còn lại của băng ghi ký hiệu trắng B, đầu đọc nhìn ký tự bên trái nhất của dãy ký tự vào và bộ điều khiển ở một trạng thái khởi đầu q_0 sau đó có thể chuyển sang các trạng thái tiếp theo. Ví dụ một mô tả hoạt động của máy Turing như hình 8-1.



Hình H8-1 - Mô tả một máy Turing

Mỗi bước chuyển của máy Turing, phụ thuộc vào ký hiệu do đầu R/W đọc được từ băng dữ liệu và trạng thái của bộ điều khiển, máy sẽ thực hiện các bước sau:

1. Ghi một ký hiệu mới trên băng tại ô đang duyệt (nghĩa là thay ký hiệu đọc được trên băng bằng ký hiệu nào đó).
2. Dịch chuyển đầu R/W (sang trái (L), sang phải (R) hoặc đứng yên ($\varnothing$)).

3. Chuyển sang trạng thái tiếp theo.

Như vậy, băng có thể được xem như vừa là kênh vào / ra vừa như là bộ nhớ vô hạn tiềm năng. Những sự khác nhau cơ bản giữa máy Turing và các loại ôtômát khác có thể mô tả vắn tắt như sau. Một ôtômát hữu hạn chỉ có thể có bộ nhớ trong được xác định bởi tập các trạng thái hữu hạn, băng vào không được dùng như bộ nhớ bổ sung. Trong ôtômát đẩy xuống, việc nhận thông tin trong bộ nhớ ngoài cũng bị hạn chế.

Do đó, rõ ràng là máy Turing là tổng quát hơn các mô hình tính toán khác mà ta đã nghiên cứu. Việc khẳng định rằng máy Turing là một mô hình toán học đủ tổng quát cho khái niệm trực giác về tính hiệu quả giải được thường được gọi là luận đề Church, được đề cập ở phần sau.

Một cách hình thức, ta định nghĩa một máy Turing như sau:

Định nghĩa 8.1. Máy Turing là một hệ thống gồm 7 thành phần MT = $(Q, \sum, \Gamma, \delta, q_0, B, F)$, trong đó:

- Q: tập khác rỗng, hữu hạn các trạng thái.
- Γ: tập khác rỗng, hữu hạn các ký tự được phép viết trên băng.
- B: ký hiệu thuộc Γ, ký hiệu trắng (Blank) trên băng.
- $\sum$: tập các ký tự đầu vào và là tập con của Γ, $B \notin \Gamma$.
- δ: hàm chuyển: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, \varnothing\}$
- $q_0 \in Q$ là trạng thái bắt đầu
- $F \subseteq Q$ là tập các trạng thái kết thúc.

Lưu ý: + $\delta()$ là hàm bộ phận, có thể không xác định đối với một số phần tử của $Q \times \Gamma$.

+ Dãy các ký tự được xem như là một kết quả tính toán (xâu đoán nhận được) của MT nếu như nó chuyển được từ trạng thái bắt đầu đạt tới trạng thái kết thúc.

8.2 Biểu diễn máy Turing

Máy Turing có thể được mô tả theo ba cách:

(i) Các mô tả hiện trạng
(ii) Bảng chuyển trạng thái
(iii) Biểu đồ (đồ thị) chuyển trạng thái.

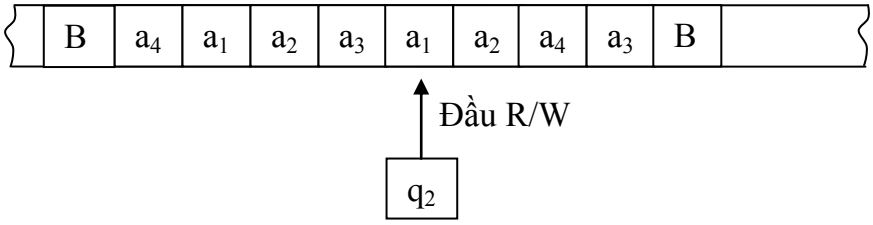
8.2.1 Biểu diễn bằng các mô tả hiện thời

Một bức ảnh chụp (Snapshot) hoạt động của MT có thể được sử dụng để mô tả về máy Turing. Những bức ảnh đó chính là các mô tả hiện thời của MT.

Định nghĩa 8.2 Một mô tả hiện trạng *(thể hiện)* (ID) của máy Turing MT là dãy $\alpha\, q\, \beta$, trong đó q là trạng thái hiện thời của MT; $\alpha\beta \in \Gamma^*$ là nội dung có trên băng, được tính từ đầu băng (những ký hiệu khác *B*) cho tới ký hiệu khác *B* (*Blank*) bên phải nhất của băng và đầu đọc đang nhìn ký tự bên trái nhất của β. Giả sử ký hiệu ở

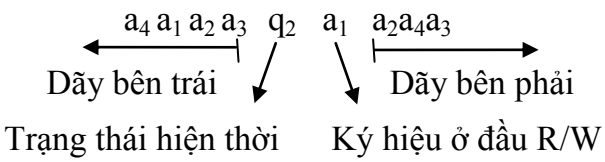
dưới đầu R/W là *a,* ký hiệu đầu tiên của *β,* thì α là dãy các ký hiệu bên trái của *a.* Nếu *β* = ε thì ký hiệu ở đầu đọc là *B*.

Ví dụ 8.1 Xét một thể hiện của một MT được cho như hình H8-2.



Hình H8-2 Một mô tả tức thời của máy Turing

Ký hiệu dưới đầu R/W là a_1 và trạng thái hiện thời của MT là q_2. Những ký hiệu khác *B* là dãy $a_4a_1a_2a_3$ được ghi ở bên trái của q_2 và những ký hiệu khác dấu *B* ở bên phải của a_1 là $a_2a_4a_3$. Vậy mô tả hiện trạng của MT trên là



Hoạt động của máy Turing M

Tương tự như ôtômát đẩy xuống, khi hàm chuyển của MT thay đổi trạng thái khi đọc một ký hiệu ở băng dữ liệu thì mô tả hiện trạng của nó cũng sẽ thay đổi theo. Tại mỗi bước, máy MT ở một trạng thái *q,* đầu đọc nhìn ký tự *s,* hoạt động của máy sẽ phụ thuộc vào bộ chuyển được xác định bởi hàm chuyển *δ().* Hoạt động này bao gồm việc ghi một ký tự mới đè lên ký tự mà đầu đọc đang nhìn, chuyển đầu đọc sang phải hay sang trái một ô, đồng thời bộ điều khiển chuyển sang một trạng thái mới.

Phép chuyển của các thể hiện của MT được định nghĩa như sau:

Định nghĩa 8.3. Giả sử $\delta(q, x_i) = (p, y, L)$ và $x_1x_2 \ldots x_{i-1}\ \mathbf{q}\ x_i \ldots x_n$ là mô tả hiện thời ID của MT. Sau khi xử lý x_i thì MT chuyển sang thể hiện

$$x_1x_2 \ldots x_{i-2}\ \mathbf{p}\ x_{i-1}y\ x_{i+1} \ldots x_n$$

- Nếu i > 1 thì sự biến đổi hiện trạng của MT được viết như sau:

$$x_1x_2 \ldots x_{i-1}\ \mathbf{q}\ x_i \ldots x_n \vdash_M x_1x_2 \ldots x_{i-2}\ \mathbf{p}\ x_{i-1}\mathbf{y}\ x_{i+1} \ldots x_n$$

- Nếu i - 1 = n thì X_i là B.

- Nếu i =1 thì không có ID kế tiếp, nghĩa là đầu đọc không được phép vượt qua cận trái của băng.

+ Tương tự, $\delta(q, x_i) = (p, y, R)$ thì thể hiện của MT được biến đổi như sau:

$$x_1x_2 \ldots x_{i-1}\ \mathbf{q}\ x_i \ldots x_n \vdash_M x_1x_2 \ldots x_{i-1}\mathbf{y}\ \mathbf{p}\ x_{i+1} \ldots x_n$$

+ Trường hợp $\delta(q, x_i) = (p, y, \varnothing)$ thực hiện như sau:

$$x_1x_2 \ldots x_{i-1}\ \mathbf{q}\ x_i \ldots x_n \vdash_M x_1x_2 \ldots x_{i-1}\ \mathbf{p}\ \mathbf{y}\ x_{i+1} \ldots x_n$$

8.2.2 Biểu diễn bằng đồ thị chuyển trạng thái

Máy Turing MT có thể biểu diễn bằng đồ thị định hướng được gắn nhãn, trong đó

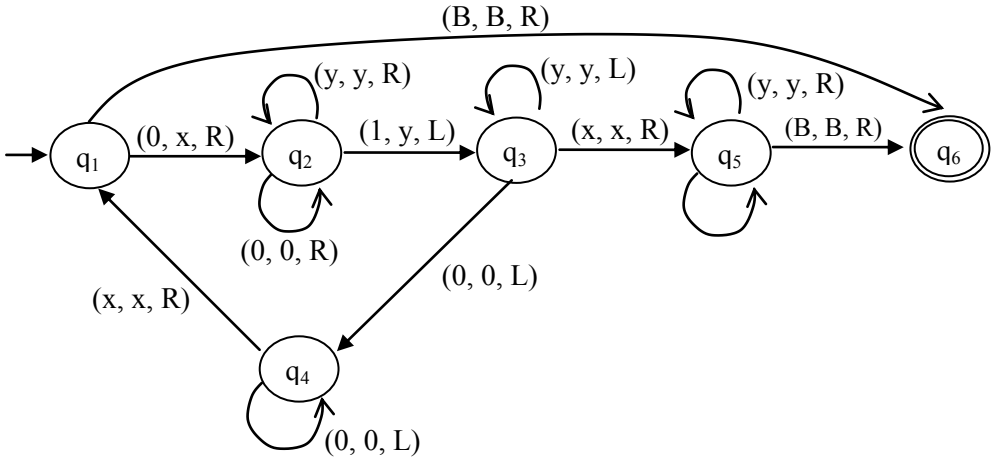
+ Tập đỉnh là tập các trạng thái,

+ Từ đỉnh q_i có cung nối với q_j và được gắn nhãn (α, β, γ) nếu

$\delta(q_i, \alpha) = (q_j, \beta, \gamma)$

+ Đỉnh bắt đầu ứng với trạng thái bắt đầu, có mũi tên đi vào và những đỉnh kết thúc được đánh dấu bằng hai vòng tròn bao nhau.

Ví dụ 8.3. Cho máy MT được biểu diễn bằng đồ thị chuyển trạng thái như hình 8.3, hãy xác định dãy tính toán của MT để xử lý đầu vào 0011.



Hình H8-3 Đồ thị chuyển trạng thái của MT

Hiển nhiên trên băng dữ liệu ta có B0011B. Trạng thái bắt đầu của MT là q_1 và ký tự ở đầu đọc là 0. Từ hình H8-3 suy ra có một cung đi từ q_1 đến q_2 với nhãn là (0, x, R). Khi đó, ký tự 0 sẽ được thay bằng x và đầu R/W chuyển sang phải một ô, đồng thời bộ điều khiển chuyển sang trạng thái q_2. Tương tự như vậy, ta có dãy tính toán như sau:

$$\underset{q_1}{\mathrm{B0011B}} \overset{(0,x,R)}{\vdash} \underset{q_2}{\mathrm{Bx011B}} \overset{(0,0,R)}{\vdash} \underset{q_2}{\mathrm{Bx011B}}$$

$$\overset{(1,x,L)}{\vdash} \underset{q_3}{\mathrm{Bx0y1B}} \overset{(0,x,L)}{\vdash} \underset{q_4}{\mathrm{Bx0y1B}} \overset{(x,x,R)}{\vdash} \underset{q_1}{\mathrm{Bx0y1B}}$$

$$\overset{(0,x,R)}{\vdash} \underset{q_2}{\mathrm{Bxxy1B}} \overset{(y,y,R)}{\vdash} \underset{q_2}{\mathrm{Bxxy1B}} \overset{(1,y,L)}{\vdash} \underset{q_3}{\mathrm{BxxyyB}}$$

$$\overset{(y,y,R)}{\vdash} \underset{q_5}{\mathrm{BxxyyB}} \overset{(B,B,R)}{\vdash} \underset{q_5}{\mathrm{BxxyyBB}}$$

$$\overset{(y,y,L)}{\vdash} \underset{q_3}{BxxyyB} \overset{(x,x,R)}{\vdash} \underset{q_5}{BxxyyB} \overset{(y,y,R)}{\vdash} \underset{q_5}{BxxyyB}$$

Chúng ta hãy xét máy Turing MT = (Q, ∑, Γ, δ, q_0, B, F). *Một tính toán của MT* với dãy ký tự vào $\omega \in \Sigma^*$, là dãy hiện trạng C_0, C_1, …, C_n, sao cho $C_0 = q_0\omega$; $C_i \vdash_{MT} C_{i+1}$, với $0 \le i < n$; và trạng thái của C_n là trạng thái kết thúc. Máy MT không đoán nhận ω nếu nó hoặc dừng ở trạng thái không được đoán nhận (trạng thái không kết thúc) hoặc nói chung là không dừng.

Như vậy, *băng vô hạn có thể xem vừa như kênh vào / ra, vừa như một bộ nhớ ngoài vô hạn tiềm năng của MT*.

Ta nói, MT *đoán nhận (chấp nhận) dãy (xâu) vào ω*, nếu dãy tính toán của MT với đầu vào ω là dừng ở hiện trạng cuối cùng có chứa trạng thái kết thúc, nghĩa là MT dừng ở hiện trạng chấp nhận.

Một cách hình thức, ta định nghĩa tập hợp các dãy được đoán nhận bởi MT là tập

$$L(MT) = \{w \mid w \in \Gamma^* \text{ và } q_0 w \vdash_{MT}^* \alpha_1 p \alpha_2 \text{ với } p \in F \text{ còn } \alpha_1\alpha_2 \in \Gamma^*\}$$

Trong đó, $\vdash_{MT}^*$ là bao đóng bắc cầu của $\vdash_{MT}$.

8.2.3 Biểu diễn bằng bảng chuyển trạng thái

Máy Turing MT có thể được biểu diễn thông qua hàm chuyển dưới dạng một bảng chuyển trạng thái, trong đó các hàng là các trạng thái và các hàng là các ký hiệu trên băng, kể cả ký hiệu trắng B.

Ví dụ 8.4. Xét máy MT được cho như trên bảng 8.2.

Bảng 8.2

Trạng thái	Ký tự trên băng				
	0	1	x	y	B
→ q_1	xRq_2				bRq_5
q_2	$0Rq_2$	yLq_3		yRq_2	
q_3	$0Lq_4$		xRq_5	yLq_3	
q_4	$0Lq_4$		xRq_1		
q_5				yRq_5	bRq_6
q_6					

Mô tả hoạt động của MT khi xử lý (**a**) 011, (**b**) 0011, (**c**) 001. MT đoán nhận được dãy nào trong ba dãy trên?

(a) Xét dãy thứ nhất: $q_1011 \vdash xq_211 \vdash q_3xy \vdash xq_5y1 \vdash xyq_51$.

Bởi vì $\delta(q_5, 1)$ không được định nghĩa và MT dừng ở trạng thái không kết thúc nên 011 không được đoán nhận bởi MT.

(b) $q_1 0011 \vdash xq_2 011 \vdash x0q_2 11 \vdash xq_3 0y1 \vdash q_4 x0y1 \vdash xq_1 0y1$

$\vdash xxq_2 y1 \vdash xxyq_2 1 \vdash xxq_3 yy \vdash xq_3 xyy \vdash xxq_5 yy$

$\vdash xxyq_5 y \vdash xxyyq_5 B \vdash xxyyBq_6$.

MT dừng ở trạng thái kết thúc là q_6, vậy 0011 là từ được MT chấp nhận.

(c) $q_1 001 \vdash xq_2 01 \vdash x0q_2 1 \vdash xq_3 0y \vdash q_4 x0y$

$\vdash xq_1 0y \vdash xxq_2 y \vdash xxyq_2$.

MT dừng ở trạng thái q_2 không phải là trạng thái kết thúc, nên 001 không đoán nhận được bởi MT.

8.3 Thiết kế máy Turing

Để xây dựng một máy Turing ta có thể thực hiện theo sự hướng dẫn dưới đây:

1. Mục tiêu của việc đọc ký hiệu của đầu R/W là cần biết xem "cái gì" máy Turing cần thực hiện ở bước tiếp theo. Máy phải ghi lại những ký hiệu đã đọc qua.
2. Số các trạng thái phải là cực tiểu. Điều này thực hiện được bằng cách chỉ thay đổi trạng thái khi có sự thay đổi ký tự cần ghi vào băng hoặc khi có sự thay đổi hướng chuyển dịch của đầu R/W.

Ví dụ 8.5. Thiết kế máy Turing MT để đoán nhận tất cả các dãy có chẵn các số 1.

MT được xây dựng như sau:

(i) Từ trạng thái bắt đầu q_1, MT đọc 1, chuyển sang q_2 và ghi B (ký hiệu trắng)

(ii) Ở trạng thái q_2, MT đọc 1, chuyển về q_1 và cũng ghi B.

(iii) q_1 cũng là trạng thái kết thúc.

Vậy, MT = $(\{q_1, q_2\}, \{1, B\}, \{1, B\}, \delta, q_1, B, \{q_1\})$, trong đó δ được định nghĩa như trong bảng 8.3.

Bảng 8.3

Trạng thái	1
$\rightarrow q_1$	Bq_2R
q_2	Bq_1R

Dễ dàng kiểm tra được 11, 1111, … là các dãy (chẵn các số 1) được MT đoán nhận và 111, 11111, …, sẽ không được MT chấp nhận.

Ví dụ 8.6. Thiết kế MT đoán nhận ngôn ngữ $L = \{ 0^n 1^n \mid n \geq 1\}$

Khởi đầu MT chứa 0^n1^n bên trái nhất trên băng sau đó là *vô hạn khoảng trống B*. MT lặp lại quá trình sau:

- MT thay 0 bên trái nhất bằng X rồi chuyển sang phải tới 1 trái nhất, MT thay chữ số 1 này bằng Y rồi dịch chuyển về bên trái cho tới khi gặp X phải nhất nó chuyển sang phải một ô (tới 0 trái nhất) rồi tiếp tục lặp một chu trình mới.
- Nếu trong khi dịch chuyển sang phải để tìm 1 mà MT gặp B thì MT dừng và không đoán nhận dãy vào. Tương tự, khi MT đã thay hết 0 bằng X và kiểm tra còn 1 trên băng thì MT cũng dừng và không đoán nhận dãy vào.
- MT đoán nhận dãy vào nếu như không còn ký hiệu 1 nào nữa trên băng.

Đặt MT = $(Q, \sum, \Gamma, \delta, q_0, B, F)$ với các thành phần:

$$Q = \{q_0, q_1, q_2, q_3, q_4\};\ \sum = \{0, 1\};\ \Gamma = \{0, 1, X, Y, B\} \text{ và } F = \{q_4\}.$$

Ta có thể hình dung mỗi trạng thái là một câu lệnh hoặc một nhóm các câu lệnh trong chương trình. Trạng thái q_0 là trạng thái khởi đầu và nó làm cho ký hiệu 0 bên trái nhất thay bằng X. Trạng thái q_1 được dùng để tiến sang phải bỏ qua các số 0 và Y để tìm 1 bên trái nhất. Nếu MT tìm thấy 1 nó thay 1 bằng Y rồi đi vào trạng thái q_2. Trạng thái q_2 đưa MT tiến sang trái cho tới X đầu tiên và đi vào trạng thái q_0, dịch chuyển sang phải để tới 0 bên trái nhất và tiếp tục một chu trình mới. Khi MT tiến sang phải trong trạng thái q_1, nếu B hoặc X được tìm thấy trước 1 thì dãy vào bị loại bỏ (không chấp nhận) vì có chứa nhiều ký hiệu 0 hơn 1 hoặc dãy vào không có dạng 0^*1^*.

Trạng thái q_0 còn có vai trò khác. Nếu trạng thái q_2 tìm thấy X bên phải nhất và ngay sau đó là Y thì các số 0 đã được xét hết, do đó ở trạng thái bắt đầu một chu trình mới q_0 không tìm thấy ký hiệu 0 nào để thay thế thành X mà chỉ gặp Y thì MT đi vào trạng thái q_3 duyệt qua các Y để kiểm tra có hay không có ký hiệu 1 còn lại. Nếu theo ngay sau các Y là B, nghĩa là trên băng nhập không còn ký hiệu 1 nào nữa thì MT sẽ đi vào q_4 (trạng thái kết thúc) để đoán nhận dãy vào. Ngược lại dãy vào bị loại bỏ.

Hàm chuyển δ được cho trong bảng sau:

Bảng 8.4

Trạng thái	Ký hiệu				
	0	1	x	y	B
→ q_0	(q_1, x, R)	-	-	(q_1, x, R)	-
q_1	(q_1, x, R)	(q_2, x, L)	-	(q_1, x, R)	-
q_2	(q_2, x, L)	-	(q_0, x, R)	(q_2, y, L)	-
q_3	-	-	-	(q_2, y, L)	$(q_4, B, \varnothing)$
q_4	-	-	-	-	-

Các phép chuyển hình thái của MT đối với dãy vào 0011:

$q_0 0011 \vdash Xq_1 011 \vdash X0q_1 11 \vdash X q_2 0Y1 \vdash q_2 X0Y1 \vdash X q_0 0Y1 \vdash XXq_1 Y1 \vdash XXY q_1 1 \vdash XX q_2 YY \vdash X q_2 XYY \vdash XX q_0 YY \vdash XXYq_3 Y \vdash XXYYq_3 \vdash XXYYq_4$.

Tương tự, chúng ta có thể kiểm tra được 0^n1^n, n = 1, 2, … được MT đoán nhận.

Ví dụ 8.7. Thiết kế MT đoán nhận ngôn ngữ $L = \{1^n2^n3^n \mid n \geq 1\}$.

Trước khi xây dựng MT, chúng ta hãy tìm thủ tục để xử lý dãy 112233. Sau khi xử lý xong, chúng ta muốn MT ở hiện trạng $BBBBBBq_7$. Thủ tục đó được thực hiện như sau:

(i) Từ trạng thái khởi đầu q_1, đầu R/W đọc và thay 1 bằng B, chuyển dịch sang phải, đồng thời chuyển sang q_2. Trong trạng thái q_2 nếu ký tự ở đầu R/W là 1 thì nó chuyển sang phải.

(ii) Đọc ký tự 2 trái nhất, đầu R/W đọc và thay 2 bằng B, chuyển dịch sang phải, đồng thời trạng thái chuyển sang q_3. Trong trạng thái q_3 nếu ký tự ở đầu R/W là 2 thì nó chuyển sang phải.

(iii) Đọc ký tự 3 trái nhất, đầu R/W đọc và thay 3 bằng B, chuyển dịch sang phải, đồng thời trạng thái chuyển sang q_4. Trong trạng thái q_4 nếu ký tự ở đầu R/W là 3 thì nó chuyển sang phải.

(iv) Sau khi đọc chữ 3 phải nhất, đầu R/W đọc chuyển dịch sang trái cho đến khi gặp chữ số 1 trái nhất.

(v) Các bước (i) – (iv) được lặp lại để lần lượt thay tất cả các chữ số 1, 2, 3 bằng B.

Dãy tính toán của MT để xử lý 112233 là:

$$q_1 112233 \vdash Bq_2 12233 \vdash B1q_2 2233 \vdash B1Bq_3 233 \vdash B1B2q_3 33$$
$$\vdash B1B2Bq_4 3 \vdash B1B2q_5 B3 \vdash B1Bq_5 2B3 \vdash B1q_5 B2B3$$
$$\vdash Bq_5 1B2B3 \vdash q_6 B1B2B3 \vdash Bq_1 1B2B3 \vdash BBq_2 B2B3$$
$$\vdash BBBq_2 2B3 \vdash BBBBq_3 B3 \vdash BBBBBq_3 3 \vdash BBBBBq_4 B$$
$$\vdash BBBBBBBq_7$$

Vậy, $q_1 112233 \vdash^* BBBBBBBq_7$. q_7 là trạng thái kết thúc nên 112233 được MT đoán nhận.

Từ đó, chúng ta có thể xây dựng MT để đoán nhận $L = \{1^n2^n3^n \mid n \geq 1\}$ có hàm chuyển được cho như trong bảng 8.5.

Bảng 8.5

Trạng thái	Ký tự trên băng			
	1	2	3	B
→ q_1	BRq_2			BRq_1
q_2	$1Rq_2$	BRq_3		BRq_2
q_3		$2Rq_3$	BRq_4	BRq_3
q_4			$3Rq_5$	BLq_7
q_5	$1Lq_6$	$2Lq_5$		BLq_5
q_6	$1Lq_6$			BRq_1
q_7				

8.4 Ngôn ngữ đoán nhận và "Hàm tính được"

Ngôn ngữ được đoán nhận bởi một máy Turing được gọi là ngôn ngữ *đệ qui đếm được* [4]. Đó là một lớp ngôn ngữ rất rộng, nó thực sự chứa ngôn ngữ phi ngữ cảnh và những ngôn ngữ không thể xác định được các thành phần một cách máy móc.

8.4.1 Máy Turing như là một máy tính hàm số nguyên

Máy Turing cũng có thể được xem như là một máy tính của các *hàm số nguyên* (đi từ tập số nguyên đến tập số nguyên). Mỗi số nguyên ta viết dưới dạng số trong *hệ nhất phân* (unary), tức là với một số $i \neq 0$ ta viết thành chuỗi 0^i (gồm i chữ số 0). Nếu hàm f() có *k* đối số $i_1, i_2, ..., i_k$ thì ta viết lần lượt các số nguyên này trên băng của MT ngăn cách nhau bởi 1, nghĩa là dãy vào có dạng $0^{i_1}\mathbf{1}0^{i_2}\mathbf{1}...\mathbf{1}0^{i_k}$. Nếu MT dừng (đoán nhận hoặc không đoán nhận dãy vào) với băng 0^m thì ta nói $f(i_1, i_2, ..., i_k) = m$.

Chú ý rằng, ta cũng có thể tính được hàm chỉ có một đối số. Nếu f() xác định với mọi bộ đối số $i_1, i_2, ..., i_k$ thì ta gọi f() là hàm *đệ qui toàn bộ*. Một hàm f() tính được bởi máy Turing ta gọi là hàm *đệ qui bộ phận*. Hàm đệ qui bộ phận tương tự như ngôn ngữ đệ qui đếm được, bởi vì nó tính được bởi máy Turing, nhưng có thể không dừng với một số đối số nào đó. Hàm đệ qui toàn bộ tương tự như ngôn ngữ đệ qui vì MT sẽ dừng trên mọi dãy vào.

Ví dụ 8.8. Thiết kế máy Turing tính toán phép trừ riêng

Ta định nghĩa *phép trừ riêng* như sau:

$$f(m, n) = m \setminus n = \begin{cases} m - n, & \text{nếu } m \geq n \\ 0, & \text{ngược lại } m < n \end{cases}$$

+ Dãy vào: $0^m\mathbf{1}0^n$

+ Dãy ra: $0^{m\setminus n}$

MT lặp lại việc thay thế lần lượt từng số 0 ở đầu băng bằng B rồi tiến sang phải, sau 1 và tìm 0 rồi thay 0 này bằng 1. MT lại chuyển sang trái cho đến khi gặp B

đầu tiên thì dừng lại, trở về trạng thái bắt đầu và tiếp tục vòng lặp như trên. MT dừng nếu:

i) Khi sang phải tìm 0 bên phải, MT gặp B. Lúc này MT đã thay n số 0 bên phải chuỗi dãy vào 0^m10^n thành 1 và n + 1 số 0 bên trái thành B, trường hợp này xảy ra khi trong chuỗi dãy vào có m > n. Do vậy, MT phải thay lại tất cả n + 1 số 1 sau thành B, và sau đó dịch trái thay trả lại một B về thành 0, cuối cùng trên băng còn lại kết quả phép trừ là m - n số 0.

ii) Khi bắt đầu một vòng lặp mới, MT không tìm thấy 0 để đổi thành B, lúc này m số 0 đầu đã bị đổi thành B, trường hợp này xảy ra khi n ≤ m. Khi đó, MT thay tất cả các số 1 và 0 trên băng thành B để cho kết quả phép trừ là 0 (biểu diễn gồm toàn ký hiệu B trong hệ nhất phân).

Ta xây dựng MT như sau: MT = $(\{q_0, q_1, ..., q_6\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_6\})$.

MT sẽ bắt đầu bằng 0^m10^n trên băng và kết thúc với $0^{m\backslash n}$ trên băng. Các phép chuyển trạng thái được định nghĩa như sau:

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

MT thay 0 đầu băng bởi B.

2) $\delta(q_1, 0) = (q_1, 0, R)$

$\delta(q_1, 1) = (q_2, 1, R)$

MT di chuyển sang phải qua 0 tìm 1.

3) $\delta(q_2, 1) = (q_2, 1, R)$

$\delta(q_2, 0) = (q_3, 1, L)$

MT di chuyển sang phải vượt qua 1 đến khi gặp 0, đổi 0 thành 1.

4) $\delta(q_3, 0) = (q_3, 0, L)$

$\delta(q_3, 1) = (q_3, 1, L)$

$\delta(q_3, B) = (q_0, B, R)$

MT dịch trái tới khi gặp B, trở về trạng thái q_0 và bắt đầu một vòng lặp mới.

5) $\delta(q_2, B) = (q_4, B, L)$

$\delta(q_4, 1) = (q_4, B, L)$

$\delta(q_4, 0) = (q_4, 0, L)$

$\delta(q_4, B) = (q_6, 0, \varnothing)$

Nếu ở trạng thái q_2 sang phải tìm 0 để thay thành 1 nhưng chỉ gặp B thì ta xét trường hợp kết thúc i) ở trên: MT đi vào trạng thái q_4 và chuyển sang trái đổi tất cả 1 thành B cho tới khi gặp một B bên trái đầu tiên. B này sẽ được thay lại thành 0 rồi MT đi vào trạng thái kết thúc q_6 và dừng.

6) $\delta(q_0, 1) = (q_5, B, R)$

$\delta(q_5, 0) = (q_5, B, R)$

$\delta(q_5, 1) = (q_5, B, R)$

$\delta(q_5, B) = (q_6, B, \varnothing)$

Nếu ở trạng thái bắt đầu vòng lặp mới q_0 gặp 1 thay vì gặp 0, thì khối các số 0 bên trái đã xét hết, đây là trường hợp kết thúc ii) nêu trên: MT sẽ đi vào trạng thái q_5, xoá phần còn lại của băng rồi đi vào trạng thái kết thúc q_6 và dừng.

Chẳng hạn MT tính toán phép trừ 2\1 (tức dãy vào 0010) như sau:

$$q_0 0010 \vdash B\, q_1 010 \vdash B0q_1 10 \vdash B01q_2 0 \vdash B0q_3 11 \vdash Bq_3 011 \vdash q_3 B011$$

$$\vdash Bq_0 011 \vdash BBq_1 11 \vdash BB1q_2 1 \vdash BB11q_2 \vdash BB1q_4 1$$

$$\vdash BBq_4 1 \vdash Bq_4 \vdash Bq_6 0.$$

Tương tự, xét tính toán 1\2 (tức dãy vào 0100):

$$q_0 0100 \vdash Bq_1 100 \vdash B1q_2 00 \vdash Bq_3 110 \vdash q_3 B110 \vdash Bq_0 110$$

$$\vdash BBq_5 10 \vdash BBBq_5 0 \vdash BBBBq_5 \vdash BBBBq_6.$$

8.4.2 Các chương trình con

Cũng giống như một chương trình máy tính hiện đại, *máy Turing có thể đóng vai trò tương tự như bất kỳ một kiểu chương trình con nào trong ngôn ngữ lập trình, bao gồm thủ tục đệ qui hoặc hàm số*. Ý tưởng chung là: ta viết một phần chương trình của MT như là một chương trình con. Nó sẽ được thiết kế như một hệ thống có chứa một trạng thái khởi đầu và một trạng thái trở về. Trạng thái trở về là trạng thái không có phép chuyển kế tiếp và nó sẽ đóng vai trò là trạng thái khởi đầu của một máy Truring khác hoặc là một trạng thái nào đó trong một máy khác. Nghĩa là từ trạng thái trở về của máy này ta tiếp tục các phép chuyển của một máy khác, sự kiện này có ý nghĩa như là lời gọi một chương trình con khác hoặc tiếp tục thực hiện chương trình cấp trên. Lưu ý, các trạng thái của chương trình con phải phân biệt với chương trình cấp trên của nó trong cấu trúc phân cấp của các chương trình con.

Ví dụ 8.9. Thiết kế MT thực hiện phép nhân 2 số nguyên m với n.

+ Dãy vào: $0^m 10^n$

+ Output: $0^{m \times n}$

MT bắt đầu với $0^m 10^n$ trên băng và kết thúc với $0^{m \times n}$ trên băng được bao quanh bởi các B.

Ý tưởng chung: là đặt thêm số 1 sau $0^m 10^n$ rồi chép khối n số 0 sang phải m lần, mỗi lần xoá một con 0 bên trái của 0^m. Ta được kết quả cuối cùng là $10^n 10^{m \times n}$. Bây giờ ta chỉ việc xoá $10^n 1$ ta sẽ được kết quả $0^{m \times n}$.

Phần chính của thuật toán trên là thủ tục COPY để chép n số 0 sang phải. Thủ tục này được xác định bằng các hàm chuyển sau:

Bảng 8.6

Trạng thái	Ký tự trên băng			
	0	1	1	B
→ q_1	$(q_2,2,R)$	$(q_4,1,L)$		
q_2	$(q_2,0,R)$	$(q_2,1,R)$		$(q_3,0,L)$
q_3	$(q_3,0,L)$	$(q_3,1,L)$	$(q_1,2,R)$	
q_4		$(q_5,1,R)$	$(q_4,0,L)$	
q_5				

Ở trạng thái q_1 nhìn thấy 0, MT đổi 0 thành 2 và đi vào trạng thái q_2. Ở trạng thái q_2, MT dịch phải tới B, ghi 0 rồi dịch trái trong trạng thái q_3. Khi ở trạng thái q_3 mà gặp 2, MT đi vào trạng thái q_1 để tiếp tục lặp lại quá trình trên cho tới khi gặp 1. Trạng thái q_4 được dùng để biến đổi 2 thành 0 và thủ tục dừng tại q_5.

Để làm đầy đủ chương trình ta phải thêm các trạng thái để biến đổi hình trạng khởi đầu $q_0 0^m 1 0^n$ thành $B0^{m-1}1q_1 0^n 1$. Tức là ta cần ba qui tắc:

$$\delta(q_0, 0) = (q_6, B, R), \delta(q_6, 0) = (q_6, 0, R), \delta(q_6, 1) = (q_1, 1, R)$$

Sau đó, ta lại thêm các phép chuyển và trạng thái cần thiết để biến đổi từ hình thái $B^i 0^{m-i} 1 q_5 0^n 1 0^{n \times i}$ thành $B^{i+1} 0^{m-i-1} 1 q_1 0^n 1 0^{n \times i}$ là trạng thái bắt đầu lại việc COPY, đồng thời kiểm tra i = m hay không (khi tất cả các 0 của 0^m đã bị xoá). Nếu i = m thì 10^n1 bị xoá và quá trình tính toán sẽ dừng ở trạng thái q_{12}.

8.5 Máy Turing không đơn định

Máy Turing trong mô hình gốc nêu trên được gọi là *máy Turing đơn định* vì hàm chuyển δ() là đơn trị. Máy Turing không đơn định có mô hình tương tự như mô hình gốc nhưng điểm khác biệt ở chỗ là trong *mỗi lần chuyển, máy Turing có thể lựa chọn một trong một số hữu hạn các trạng thái kế tiếp* (hàm chuyển δ() là không đơn trị)*, lựa chọn hướng chuyển đầu R/W, và lựa chọn ký hiệu ghi vào băng thay thế ký hiệu vừa đọc được*.

Lưu ý: máy Turing không đơn định vốn không dự định để mô hình hoá các hoạt động tính toán. Nó chỉ đơn thuần là một máy toán học bổ trợ và có thể hình dung như một máy để kiểm chứng một phỏng đoán có đúng hay không.

8.6 Luận đề Church

*Giả thuyết cho rằng khái niệm trực giác "**Hàm tính được**" có thể được định nghĩa bằng lớp các hàm đệ quy bộ phận là giả thuyết Church hay còn được gọi là luận đề Church - Turing.* Chúng ta không thể hy vọng chứng minh được giả thuyết Church cũng như những định nghĩa không hình thức về "*sự tính được*", nhưng chúng ta có thể cho những dẫn chứng về những khả năng của chúng. Trong một thời gian dài, khái niệm trực giác về "sự tính được" đặt không giới hạn trên số bước hoặc tổng số các phép lưu trữ, có vẻ như các hàm đệ quy bộ phận có thể tính

được một cách trực giác, mặc dù cũng có một số hàm không thể tính được, trừ khi ta đặt giới hạn cho việc tính toán sau đó hoặc ít nhất thiết lập được liệu có hay không có phép tính cuối cùng.

Điều còn không rõ là, liệu lớp các hàm đệ quy bộ phận có thể bao hàm tất cả mọi "hàm tính được" hay không. Những nhà logic học đã đưa ra nhiều công thức khác, chẳng hạn như *phép tính-λ, hệ thống Post* và các *hàm đệ quy tổng quát*. Tất cả chúng định nghĩa cùng một lớp hàm, cụ thể là hàm đệ quy bộ phận. Hơn nữa, các mô hình máy tính trừu tượng, chẳng hạn như mô hình *RAM* cũng được xem xét như một hàm đệ quy bộ phận.

Mô hình RAM bao gồm một số vô hạn các từ nhớ, đánh số 0, 1, ..., mỗi một từ nhớ có thể lưu giữ một số nguyên bất kỳ và một số hữu hạn các thanh ghi số học cũng có khả năng giữ các số nguyên bất kỳ. Các số nguyên có thể được giải mã thành các dạng thông thường của các chỉ thị máy tính. Chúng ta sẽ không định nghĩa mô hình RAM một cách hình thức hơn, nhưng sẽ rõ ràng hơn nếu chúng ta chọn một tập các chỉ thị phù hợp. RAM mô phỏng mọi máy tính hiện có. Chứng minh rằng mô hình máy Turing cũng có khả năng tương đương như mô hình RAM được chỉ ra dưới đây hay có thể nói: một máy Turing cũng có sức mạnh như một máy RAM.

Mô phỏng mô hình RAM bởi máy Turing

Định lý 8.1: *Một máy Turing MT có thể mô phỏng một RAM, với điều kiện là mỗi chỉ thị RAM cũng có thể được mô phỏng bởi một MT.*

8.7 Sự tương đương giữa văn phạm loại 0 và máy Turing

Họ văn phạm rộng lớn nhất theo sự phân loại của Noam Chomsky là những văn phạm có các qui tắc dạng $\alpha \rightarrow \beta$, với α, β là các chuỗi tùy ý chứa ký hiệu văn phạm sao cho $\alpha \neq \Lambda$. Lớp văn phạm này được biết như là văn phạm *loại 0*, văn phạm *ngữ cấu* hay văn phạm *không hạn chế*. Trong phần này chúng ta xét mối quan hệ tương đương giữa văn phạm loại 0 và máy Turing.

Ví dụ 8.10. Cho một văn phạm không hạn chế sinh ra ngôn ngữ

$L = \{a^i \mid i$ là lũy thừa của $2\}$ với tập qui tắc như sau:

$G = (\{S, A, B, C, D, E\}, \{a, \Lambda\}, P, S)$

Với P = {
1. $S \rightarrow ACaB$
2. $Ca \rightarrow aaC$
3. $CB \rightarrow DB$
4. $CB \rightarrow E$
5. $aD \rightarrow Da$
6. $AD \rightarrow AC$
7. $aE \rightarrow Ea$
8. $AE \rightarrow \Lambda$ }

Trong văn phạm trên, biến A và B giữ vai trò là ký hiệu đánh dấu cận trái và cận phải của một chuỗi thuộc ngôn ngữ. C di chuyển từ trái sang phải qua chuỗi các ký

hiệu a nằm giữa hai biến A và B, và gấp đôi số ký hiệu a đó lên theo qui tắc (2). Khi C chạm đến cận phải B, nó sẽ thay thế thành D hay E theo qui tắc (3) hoặc (4). Nếu D được chọn thay thế thì D lại quay về trái theo qui tắc (5), cho đến khi gặp cận trái A thì thay thế lại thành C theo qui tắc (6) và cho phép lặp lại chu trình. Còn nếu E được chọn để thay thế, thì theo qui tắc (4), B sẽ biến mất, sau đó E quay về trái theo qui tắc (7) cho đến khi gặp cận trái A thì xóa A và mất đi theo qui tắc (8), cho ra chuỗi có dạng 2^i ký hiệu a, với i > 0.

Có thể chứng minh quy nạp theo số bước dẫn xuất: nếu qui tắc (4) chưa được dùng đến thì chuỗi trong dẫn xuất có một trong ba dạng như sau:

(i) S

(ii) Aa^iCa^jB, với i + 2j là một lũy thừa của 2.

(iii) Aa^iDa^jB, với i + j là một lũy thừa của 2.

Khi qui tắc (4) được áp dụng thì ta sẽ có chuỗi dạng Aa^iE, trong đó i là một lũy thừa của 2. Sau đó, ta chỉ có thể áp dụng i lần qui tắc (7) để đi tới dạng câu AEa^i. Cuối cùng, với qui tắc (8), ta thu được chuỗi dạng a^i với i là lũy thừa của 2.

Phần tiếp theo, chúng ta sẽ xét mối tương quan giữa văn phạm không hạn chế này và mô hình máy Turing. Chúng ta chứng minh hai định lý dưới đây thể hiện mối tương quan giữa lớp văn phạm không hạn chế và lớp ngôn ngữ đệ quy đếm được – lớp ngôn ngữ được đoán nhận bởi một máy Turing. Định lý đầu tiên sẽ chứng tỏ rằng mọi ngôn ngữ loại 0 phát sinh một tập đệ quy *đếm được*. Và sau đó ta sẽ xây dựng một thuật toán để đếm được tất cả các chuỗi thuộc văn phạm loại 0.

Định lý 8.2: Nếu L là ngôn ngữ sinh bởi một văn phạm loại 0 $G = (V_N, \Sigma, P, S)$ thì L là ngôn ngữ đệ quy đếm được.

Chứng minh:

Thiết lập một máy Turing MT không đơn định, hai băng đoán nhận ngôn ngữ L. Băng thứ nhất (băng 1) của MT chứa chuỗi vào w, còn băng thứ hai (băng 2), máy phát sinh các dạng chuỗi α của G. Đầu tiên, chuỗi α được phát sinh trên băng 2 là ký hiệu bắt đầu S. Sau đó, MT lặp lại quá trình sau:

(i) Chọn một cách ngẫu nhiên một vị trí i trên α với $1 \le i \le |\alpha|$, nghĩa là MT xuất phát từ bên trái chuỗi α rồi tùy chọn giữa hai khả năng: hoặc chọn i là vị trí hiện tại, hoặc dịch chuyển sang phải và lặp lại quá trình.

(ii) Chọn một qui tắc $\beta \to \gamma$ trong số các qui tắc thuộc tập qui tắc của G.

(iii) Nếu chuỗi con β xuất hiện trong α kể từ vị trí thứ i, MT thay thế chuỗi β bởi γ (dĩ nhiên nếu $|\beta| \ne |\gamma|$ thì phải dịch chuyển phần cuối của α để đủ chỗ trống cần cho phép thay thế).

(iv) So sánh chuỗi phát sinh được với chuỗi nhập w trên băng 1. Nếu giống nhau thì chuỗi mới phát sinh sẽ được chấp nhận. Nếu khác nhau thì MT trở về bước (i). Ta có thể chứng minh được rằng tất cả và chỉ có những chuỗi thuộc G mới xuất hiện trên băng 2 ở bước (iv).

Vậy L(MT) = L(G) = L. ■

Định lý 8.3: Nếu L là ngôn ngữ đệ quy đếm được thì L = L(G), với một văn phạm không hạn chế G (loại 0) nào đó.

Chứng minh:

Giả sử ngôn ngữ L được đoán nhận bởi máy Turing MT = (Q, ∑, Γ, δ, q_0, B, F). Ta sẽ xây dựng một văn phạm không hạn chế G mà mỗi chuỗi dẫn xuất của nó phát sinh theo ba bước như sau:

(i) G phát sinh một cách ngẫu nhiên một chuỗi w thuộc Σ. Chuỗi này được viết thành hai bản: một sẽ lưu giữ cho đến khi kết thúc, một sẽ thay đổi trong quá trình làm việc của MT.

(ii) G mô phỏng lại quá trình làm việc của của MT trên chuỗi w, bằng cách lặp lại đúng quá trình làm việc của MT.

(iii) Khi bước (ii) kết thúc, với sự xuất hiện của một trạng thái kết thúc q ∈ F của MT (nghĩa là chuỗi w đã được MT chấp nhận). Lúc đó G tiếp tục thu giảm để chuyển dạng câu đã có về như chuỗi w. Và như vậy, có nghĩa là chuỗi w đã được G sinh ra.

Một cách hình thức, ta thiết lập văn phạm G = (V_N, Σ, P, S_1)

Với V_N = ((Σ ∪ { ε }) × Γ) ∪ { S_1, S_2, # })

Và tập qui tắc P được xây dựng như sau:

1. a) $S_1 \to \#q_0 S_2\#$

   b) $S_2 \to [a, a] S_2\#$, ∀a ∈ ∑

   c) $S_2 \to \varepsilon$

- Nếu δ(q, X) = (p, Y, R) với p, q ∈ ∑; X, Y ∈ Γ thì thêm các qui tắc dạng (2.a) và (2.b) sau đây vào tập qui tắc P:

2. a) q[a, X][b, Z] → [a, Y]p[b, Z], ∀a, b ∈ Σ ∪ {ε} và ∀Z ∈ Γ

   b) q[a, X]# → [a, Y]p[ε, B], ∀a ∈ Σ ∪ {ε}

- Nếu δ(q, X) = (p, Y, L) với p, q ∈ ∑; X, Y ∈ Γ thì thêm các qui tắc dạng (2.c) sau đây vào tập qui tắc P:

   c) [b, Z]q[a, X] → q[b, Z]p[a, Y], ∀a, b ∈ Σ ∪ {ε} và ∀Z ∈ Γ

- Nếu q ∈ F thì thêm các qui tắc (3.a-e) sau đây vào tập qui tắc P:

3. a) [a, X]q → qap, ∀a ∈ Σ ∪ {ε} và ∀X ∈ Γ

   b) q[a, X] → qap, ∀a ∈ Σ ∪ {ε} và ∀X ∈ Γ

   c) q# → ε

   d) #q → ε

e) $q \rightarrow \varepsilon$

Dùng các qui tắc (1.a-c), ta có dãy dẫn xuất:

$$S_1 \Rightarrow_G^* \#q_0\,[a_1, a_1][a_2, a_2] \ldots [a_n, a_n]\#$$

Dãy dẫn xuất này thể hiện hình trạng bắt đầu của MT là: $\#q_0a_1a_2 \ldots a_n\#$. Bắt đầu từ bước này các quy tắc (2.a-c) được áp dụng. Lưu ý rằng các qui tắc này trong G phản ánh các quy tắc chuyển trạng thái đã được thiết kế cho MT. Cho nên quá trình dẫn xuất lại trong G sẽ mô phỏng lại các bước chuyển hình thái trong quá trình làm việc của MT. Nếu quá trình đó chuyển đến một trong những trạng thái kết thúc $q \in F$, tương ứng với trường hợp MT đoán nhận chuỗi $a_1a_2 \ldots a_n$, thì trong văn phạm G các quy tắc (3.a-e) sẽ được áp dụng tiếp theo và cho phép G dẫn xuất ra chính chuỗi nhập $a_1a_2 \ldots a_n$. Hay ta có: $S \Rightarrow_G^* a_1a_2 \ldots a_n$

Phần chứng minh $L(MT) \subseteq L(G)$ và $L(G) \subseteq L(MT)$ xem như bài tập. ■

8.8 Ôtômát tuyến tính giới nội và văn phạm cảm ngữ cảnh

8.8.1 Ôtômát tuyến tính giới nội

Định nghĩa 8.4: *Ta gọi ôtômát tuyến tính giới nội là một máy Turing không đơn định và không có khả năng nới rộng vùng làm việc ra khỏi mút trái và mút phải của chuỗi vào*. Ký hiệu một ôtômát tuyến tính giới nội là LBA.

Nó phải thỏa hai điều kiện sau:

1) Bộ chữ cái vào của nó có chứa thêm hai ký hiệu đặc biệt ⊄ và $ dùng làm ký hiệu đánh dấu mút trái và mút phải.

2) LBA không thực hiện phép chuyển sang trái (L) từ ⊄ và không thực hiện phép chuyển sang phải (R) từ $, và cũng không viết các ký hiệu khác lên ⊄ và $.

LBA đơn giản là một máy Turing, nhưng thay vì sử dụng một băng không giới hạn cho việc tính toán, nó bị hạn chế chỉ trong phạm vi băng chứa chuỗi vào x với hai ô chứa các ký hiệu đánh dấu cận đầu mút. Sự giới hạn này làm cho việc tính toán phải thông qua một số các hàm tuyến tính trên độ dài chuỗi, do đó ta gọi mô hình này là *ôtômát tuyến tính giới nội*. LBA không dùng các ô trống ở trên băng về phía trái và phía phải của chuỗi vào, vì vậy ký hiệu khoảng trắng B như đã dùng ở máy Turing là không cần dùng ở đây. Trái lại, để LBA nhận biết được giới hạn bên trái và giới hạn bên phải của chuỗi vào, ta phải đưa thêm vào bộ chữ cái vào S hai ký hiệu đặc biệt như ⊄, $ để đánh dấu mút trái và mút phải của chuỗi. Vậy, tại thời điểm bắt đầu, chuỗi vào ở trên băng sẽ có dạng ⊄ w $, trong đó $w \in (\Sigma - \{⊄, \$\})^*$ là chuỗi cần đoán nhận. Trong quá trình làm việc, khi đầu đọc duyệt tới ô có chứa ⊄ hay $, thì phép chuyển tiếp theo sau đó chỉ có thể là đổi trạng thái, chuyển đầu đọc trở lại phía trong phạm vi băng (tức chuyển sang phải khi gặp ⊄ và chuyển sang trái khi gặp $), và không được phép viết ký hiệu gì khác trên băng tại ô đang đọc khi gặp ⊄ và $. Vậy, LBA có thể định nghĩa một cách hình thức hơn như sau.

Định nghĩa 8.5: LBA là một hệ thống MT = (Q, Σ, Γ, δ, q_o, ⊄, \$, F), trong đó các thành phần Q, Σ, Γ, q_o, F vẫn như đã định nghĩa ở máy Turing, còn ⊄, \$ ∈ Σ và hàm chuyển:

$$\delta: Q \times \Gamma \rightarrow (Q \times \Gamma \times \{L, R\})$$

phải thỏa mãn điều kiện:

- Nếu (p, Y, E) ∈ δ(q, ⊄) thì Y = ⊄ và E = R
- Nếu (p, Y, E) ∈ δ(q, \$) thì Y = \$ và E = L

Ngôn ngữ được đoán nhận bởi LBA

Ta định nghĩa ngôn ngữ L(MT) được đoán nhận bởi LBA MT là tập hợp:

$$L(MT) = \{ w \mid w \in (\Sigma - \{⊄, \$\})^* \text{ và } q_o ⊄ w\$ \vdash_{MT}^* \alpha q \beta \text{ với } q \in F \text{ và } \alpha\beta \in \Gamma^* \}$$

Chú ý rằng các ký hiệu đánh dấu hai đầu mút ngay từ hình trạng bắt đầu, chúng đã có mặt trên băng nhập, nhưng chúng không được xem như thuộc một phần của chuỗi được đoán nhận hay không được đoán nhận bởi LBA. Vì đầu đọc của LBA không thể dịch chuyển ra ngoài phần chuỗi nhập nên chúng ta không cần định nghĩa các khoảng trống (ký hiệu B) phía bên phải của \$.

8.8.2 Văn phạm cảm ngữ cảnh (CSG)

Văn phạm cảm ngữ cảnh, như đã trình bày ở các chương trước, là một văn phạm G = (V_N, Σ, P, S), trong đó:

1) V_N là một tập hữu hạn các biến hay ký hiệu không kết thúc.
2) Σ là một tập hữu hạn các ký hiệu cuối, $V_N \cap \Sigma = \varnothing$
3) P là tập hữu hạn các qui tắc dạng $\alpha \rightarrow \beta$ trong đó $\alpha, \beta \in (V_N \cup \Sigma)^*$, chuỗi α phải có chứa biến và ràng buộc $|\alpha| \le |\beta|$
4) $S \in V_N$ là ký hiệu bắt đầu.

Ngôn ngữ do văn phạm cảm ngữ cảnh G sinh ra là

$$L(G) = \{w \mid w \in \Sigma^* \text{ và } S \overset{*}{\Rightarrow} w\}$$

L(G) được gọi là *ngôn ngữ cảm ngữ cảnh*. Thuật ngữ "cảm ngữ cảnh" có xuất xứ từ một dạng chuẩn của văn phạm dạng này, trong đó mỗi qui tắc có dạng $\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$ với $\beta \neq \Lambda$, cho thấy một biến A chỉ có thể được thay thế bởi một chuỗi β (khác rỗng) trong "ngữ cảnh" α_1 - α_2. Điều đó không giống như trong văn phạm phi ngữ cảnh, với các qui tắc có dạng $A \rightarrow \beta$ ($|\beta| \ge 0$), sự thay thế này không đòi hỏi ngữ cảnh.

Ví dụ 8.11 Xét văn phạm G = (V_N, Σ, P, S) với V_N ={ S, B, C}, ∑ ={a, b, c} và P gồm các qui tắc như sau:

1. $S \rightarrow aSBC$
2. $S \rightarrow aBC$

3. $CB \to BC$
4. $aB \to ab$
5. $bB \to bb$
6. $bC \to bc$
7. $cC \to cc$

Một cách phi hình thức, áp dụng một số qui tắc cho các chuỗi dẫn xuất sinh ra ngôn ngữ, ta dễ thấy rằng văn phạm G sinh ra ngôn ngữ có dạng:

$$L = \{a^n b^n c^n \mid n \geq 1\}$$

Thật vậy, với qui tắc (1) và (2) ta có chuỗi dẫn xuất $S \overset{*}{\Rightarrow} a^n(BC)^n$. Sau đó, bằng cách áp dụng qui tắc (3), mọi biến B sẽ được thay thế lên trước các biến C trong chuỗi dẫn xuất: $a^n(BC) \Rightarrow^* a^n B^n C^n$. Bởi qui tắc (4) và (5), mọi biến B sẽ được thay thế thành các ký hiệu kết thúc b, và cuối cùng với (6) và (7), mọi biến C cũng sẽ được thay thế thành c. Tóm lại, ta có chuỗi dẫn xuất như sau:

$$S \overset{*}{\Rightarrow} a^n(BC)^n \overset{*}{\Rightarrow} a^n B^n C^n \overset{*}{\Rightarrow} a^n b^n c^n.$$

Bài toán thành viên với văn phạm cảm ngữ cảnh

Định lý 8.4 Với mọi văn phạm cảm ngữ cảnh $G = (V_N, \Sigma, P, S)$, tồn tại một thuật toán để xác định xem một chuỗi vào $w \in \Sigma^*$ có thuộc ngôn ngữ L(G) hay không.

Chứng minh:

Giả sử $|w| = n$. Ta lập đồ thị mà mỗi đỉnh là một chuỗi thuộc $(V_N \cup \Sigma)^*$ có độ dài nhỏ hơn hoặc bằng n, có một cung từ đỉnh α đến đỉnh β nếu $\alpha \Rightarrow \beta$. Như vậy, một đường đi trong đồ thị đó tương ứng với một suy dẫn trong G. Vậy $w \in L(G)$ khi và chỉ khi có một đường đi từ đỉnh bắt đầu S tới đỉnh w trong đồ thị. Dùng bất cứ thuật toán thuật nào cho phép tìm đường nối hai đỉnh trong đồ thị (đã có nhiều thuật toán như thế ([2], [6])), ta sẽ xác định được phải chăng đã có đường đi từ đỉnh S tới đỉnh w. ■

Ví dụ 8.12: Xét văn phạm cảm ngữ cảnh $G = (V_N, \Sigma, P, S)$ với các qui tắc được cho như trong thí dụ 8.11 trên và xét chuỗi vào w = abbc. Ta cần xác định xem liệu chuỗi $w \in L(G)$?

Để tìm đường đi từ đỉnh S tới đỉnh abbc trong đồ thị nói trên ta có thể dùng phương pháp "*vết dầu loang*" như sau:

Lập các R(i), i = 0, 1, 2, … theo quy tắc sau:

$$R(0) = \{ S \}$$

$$R(i) = R(i-1) \cup \{\beta \mid \alpha \Rightarrow \beta \text{ với } \alpha \in R(i-1) \text{ và } |\beta| \leq |w|\}$$

Do $R(0) \subseteq R(1) \subseteq \ldots \subseteq R(i) \subseteq R(i+1) \subseteq \ldots \subseteq$ tập các đỉnh, vậy tồn tại số k nào đó sao cho:

$$R(k) = R(k+1) = R(k+2) = \ldots$$

Do đó, quá trình thành lập các R(i) sẽ có thể ngừng sau k bước.

Và $w \in L(G)$ khi và chỉ khi có $i \le k$ để cho $w \in R(i)$.

Trong ví dụ trên, giả sử khi ta xét $|w| = 4$, ta có:

$R(0) = \{S\}$

$R(1) = \{S, aSBC, aBC\}$

$R(2) = \{S, aSBC, aBC, abC\}$

$R(3) = \{S, aSBC, aBC, abC, abc\}$

$R(4) = R(3)$

Vậy chuỗi abbc không thuộc L(G).

8.8.3 Sự tương đương giữa LBA và CSG

Chúng ta chú ý rằng, LBA có thể đoán nhận các xâu rỗng Λ, còn CSG không thể sinh ra xâu rỗng. Ngoài trường hợp đó ra thì LBA sẽ đoán nhận chính xác tất cả các xâu được sinh ra từ CSG.

Định lý 8.5 Nếu L là một ngôn ngữ cảm ngữ cảnh thì L sẽ được đoán nhận bởi một LBA nào đó.

Chứng minh:

Cách chứng minh định lý này cũng tương tự như cách chứng minh của định lý ở chương trước về sự tương đương giữa lớp ngôn ngữ sinh từ văn phạm loại 0 với lớp ngôn ngữ mà máy Turing chấp nhận, chỉ khác là ở đây không cần dùng một băng nhập thứ hai để phát sinh các dạng câu theo xâu dẫn xuất lần lượt theo các suy dẫn của văn phạm, mà chỉ cần dùng băng thứ hai trên băng nhập của LBA vào việc đó.

Cho $G = (V_N, \Sigma, P, S)$ là một văn phạm cảm ngữ cảnh, ta xây dựng ôtômát LBA MT như sau: Băng nhập của LBA gồm hai băng: băng 1 chứa xâu nhập vào w với các ký hiệu đánh dấu ⊄, $ ở hai đầu, băng 2 dùng để phát sinh các dạng câu α. Trạng thái bắt đầu, nếu w = Λ thì MT ngừng và không đoán nhận dãy vào, nếu không thì đầu đọc viết ký hiệu S ở băng 2, ngay dưới ký hiệu bên trái nhất của xâu w, tiếp đó MT thực hiện quá trình sau:

1) Chọn trong số không đơn định một xâu con β của xâu α trên băng 2 sao cho $\beta \to \gamma$ là một qui tắc trong P.

2) Thay β bởi γ, nếu cần thiết ta phải dịch chuyển phần cuối xâu sang phải cho đủ chỗ, tuy nhiên nếu dịch chuyển ra ngoài $ thì LBA ngừng và không chấp nhận.

3) (Hình thái hiện tại ở băng 1 là ⊄w$, còn ở băng 2 là xâu α, mà $S \Rightarrow_G \alpha$ và $|\alpha| \le |w|$). So sánh băng 1 và băng 2, nếu α = w thì LBA ngừng và đoán nhận w. Nếu không thì trở về bước (1).

Như vậy khi MT đoán nhận xâu w, thì $S \overset{*}{\Rightarrow} w$. Ngược lại nếu $S \overset{*}{\Rightarrow} w$ thì mọi dạng câu α xuất hiện trong dãy dẫn xuất đó đều thoả mãn $|\alpha| \le |w|$, bởi vì mọi qui tắc $\beta \to \gamma$ trong văn phạm G đều thỏa $|\beta| \le |\gamma|$. Như vậy MT có thể thực hiện dãy dẫn xuất đó trên băng 2, giữa hai ký hiệu đánh dấu đầu mút $\not\subset$ và \$. Vậy MT đoán nhận xâu nhập w.

Tóm lại MT sẽ đoán nhận mọi xâu sinh ra bởi văn phạm G. ■

Định lý 8.6 Nếu L = L(MT) với một LBA MT = $(Q, \Sigma, \Gamma, \delta, q_o, \not\subset, \$, F)$ thì $L - \{\Lambda\}$ là một ngôn ngữ cảm ngữ cảnh.

Chứng minh:

Ta xây dựng G theo 3 giai đoạn:

(i) *Giai đoạn 1*: Văn phạm cho phép sinh ra một xâu w (xâu nhập của MT), cũng được chứa trong $\not\subset$, \$ và q_0.

(ii) *Giai đoạn 2*: Văn phạm lặp lại công việc của MT.

(iii) *Giai đoạn 3*: Khi xuất hiện trạng thái $q \in F$, ta thu về xâu w với lưu ý rằng các qui tắc $\alpha \to \beta$ đều có $|\alpha| = |\beta|$.

Quá trình mô phỏng lại các qui tắc đó bởi các qui tắc của G sẽ không có gì vướng mắc. Chỉ ở giai đoạn 3, việc xoá đi các ký hiệu đánh dấu hai đầu mút $\not\subset$ và \$, q không được phép làm rút ngắn xâu nhập lại. Để giải quyết vướng mắc này, ta gắn các ký hiệu $\not\subset$, \$, q kề bên với các ký hiệu của xâu nhập mà không để đứng rời ra như trước.

Cụ thể, giai đoạn 1 thực hiện bởi các qui tắc trong G sau:

$S_1 \to [a, q_0 \not\subset a]S_2 \qquad S_1 \to [a, q_0 \not\subset a\$]$

$S_2 \to [a, a]S_2, \qquad S_2 \to [a, a\$]$

$\forall a \in \Sigma - \{\not\subset, \$\}$

Các qui tắc trong G cho phép thực hiện giai đoạn 2, giống như LBA MT thực hiện (sinh viên tự xây dựng xem như bài tập).

Cuối cùng, ở giai đoạn 3, các qui tắc sau đây sẽ được sử dụng, với $q \in F$:

$[a, \alpha q \beta] \to a$

$\forall a \in \Sigma - \{\not\subset, \$\}$ và $\forall \alpha, \beta$ có thể có.

Chú ý rằng số qui tắc là hữu hạn, vì α và / hoặc β chỉ gồm $\not\subset$, \$ và một ký hiệu nhập vào. Chúng ta cũng có thể xoá thành phần thứ hai của một biến nếu nó liền kề với ký hiệu kết thúc bằng cách dùng các qui tắc dạng:

$[a, \alpha]b \to ab$

$b[a, \alpha] \to ba$

$\forall a, b \in \Sigma - \{\not<, \$\}$ và $\forall \alpha$ có thể có.

Như vậy các qui tắc vừa được xây dựng mô tả văn phạm là G và có thể chứng minh L(MT) - {Λ} = L(G). ■

8.9 Bài toán dừng của máy Turing

8.9.1 Những bài toán không quyết định được

Trong khoa học tính toán, chúng ta thường gặp những bài toán mà câu trả lời là "có" (yes) hoặc "không" (no), "đúng" (true) hoặc "sai" (false), được gọi là loại bài toán quyết định. Đối với một bài toán quyết định, có những bộ dữ liệu vào có câu trả lời (đầu ra) là "có" và cũng có những bộ dữ liệu vào có câu trả lời "không". Sau đây là một số bài toán quyết định liên quan đến các ngôn ngữ (hình thức, ngôn ngữ lập trình):

1. Có thể viết được chương trình tìm tất cả các sô nguyên dương x, y, z và số nguyên n > 2 để $x^n + y^n = z^n$. Bài toán đặt ra là chương trình này có dừng hay không nếu và khi nào tìm thấy lời giải? (Định lý cuối cùng của Fermat).
2. Bài toán SAT (Formula SATisfiability). Cho trước công thức Boole, hỏi có một bộ giá trị của các biến để công thức đó luôn nhận giá trị đúng hay không?
3. Đối với một ngôn ngữ lập trình bất kỳ, có xác định được hay không:
   (a) Một chương trình cho trước có thể lặp vô hạn với dữ liệu vào bất kỳ?
   (b) Một chương trình cho trước có thể cho ra kết quả?
   (c) Một chương trình cho trước có thể dừng với dữ liệu vào cho trước?
4. Đối với các ngôn ngữ hình thức, có xác định hay không:
   (a) Bài toán tương đương của các văn phạm phi ngữ cảnh?
   (b) Bài toán rỗng của ngôn ngữ cảm ngữ cảnh?
   (c) Bài toán thành viên của văn phạm loại 0, một xâu cho trước có thuộc một ngôn ngữ loại 0?
5. Bài toán dừng của máy Turing?
6. Bài toán **P = NP**? (được đề cập chi tiết ở phần sau).

…

Định nghĩa 8.6: Lớp các bài toán với hai kết quả (có hoặc không) được gọi là *quyết định được (giải được)* nếu tồn tại một thuật toán xác định và luôn dừng với kết luận (có hoặc không) đối với mọi bộ dữ liệu vào. Ngược lại là lớp các bài toán *không quyết định được* (không giải được).

Tất cả các bài toán nêu trên là không quyết định được. Sau đây chúng ta chứng minh rằng bài toán dừng của máy Turing là không giải được.

Theo giả thuyết của Church, máy Turing được xem như một hệ thống tính toán tổng quát nhất. Từ đó suy ra, việc tìm ra *một thuật toán để giải bài toán quyết định tương đương với việc xây dựng máy Turing để thực hiện thuật toán đó.*

Trước khi chứng minh định lý về tính dừng của máy Turing, chúng ta cần lưu ý thêm một số biểu diễn của máy Turing. Hàm chuyển $\delta()$ có thể biểu diễn thành bộ 5, ví dụ: $\delta(q, a) = (\gamma, \alpha, \beta)$ được biểu diễn thành $(q, a, \alpha, \beta, \gamma)$. Tất cả các trạng thái và các bộ 5-phần tử như trên có thể mã hoá thành các xâu (dãy ký tự) trên bảng chữ Σ (thường $\Sigma = \{0, 1\}$, vậy MT có thể biểu diễn bằng các dãy nhị phân). Một tập các dãy mã hoá MT như thế, sẽ được ký hiệu là d(MT). Cặp (MT, x), trong đó MT là máy Turing và x là dãy vào, được mã hoá thành d(MT) * x.

Định lý 8.7 (Định lý **Turing**) Bài toán dừng của máy Turing trên bảng chữ $\Sigma = \{0, 1\}$ là không quyết định được, nghĩa là một máy Turing bất kỳ có dừng hay không đối với dãy vào x bất kỳ trong Σ là không quyết định được.

Chứng minh:

Chúng ta chứng minh định lý bằng phương pháp phản chứng. Giả sử MT là một máy Turing bất kỳ và d(MT) là dãy mã hoá nhị phân biểu diễn cho MT. Khi đó, cặp (MT, x), x là bất kỳ, sẽ được mã hoá thành d(MT)*x. Giả sử bài toán dừng của MT đối với *x* là quyết định được, thì sẽ tồn tại thuật toán **P** để trả lời xem MT có dừng hay không. Nghĩa là,

(a) Nếu MT dừng đối với dãy vào *x* thì **P** đạt tới hiện trạng được chấp nhận,

(b) Nếu MT không dừng đối với dãy vào *x* thì **P** đạt tới hiện trạng không được chấp nhận.

Chúng ta hãy xây dựng thuật toán **Q** mới dựa vào **P** như sau:

(c) Xem d(MT) là dãy vào và sao đúp thành d(MT) *d(MT), sau đó áp dụng thuật toán **P** đối với dãy vào d(MT) *d(MT),

(d) **Q** lặp lại vô hạn nếu **P** đạt đến hiện trạng được đoán nhận và **Q** dừng nếu **P** đạt đến hiện trạng không được chấp nhận.

Theo giả thuyết Church, tồn tại một máy Turing MT′ để thực hiện **Q**. Bởi vì thuật toán **P**, cũng như **Q** thực hiện được trên mọi máy, vậy **Q** thực hiện được trên máy MT và **P** thực hiện được trên MT′. Ta có thể chọn MT′ = MT.

+ Từ (d) và (a) suy ra MT′ lặp vô hạn nếu nó dừng.

+ Từ (d) và (b) suy ra MT′ dừng nếu nó lặp vô hạn.

Nghĩa là, MT′ dừng khi và chỉ khi MT′ lặp vô hạn. Đây rõ ràng là mâu thuẫn, vậy tính dừng của máy Turing là không quyết định được. ■

Hệ quả 8.1*:* Bài toán tồn tại một máy Turing để quyết định xem có dừng hay không khi thao tác trên một dãy vào bất kỳ là không giải được.

Hệ quả 8.2*:* Tồn tại máy Turing MT trên {0, 1} và một trạng thái q_m sao cho không tồn tại thuật toán để quyết định xem MT có đạt đến q_m từ một hiện trạng nào đó hay không.

8.9.2 Bài toán về sự tương ứng của Post

Bài toán về sự tương ứng Post PCP (Post Correspondence Problem) được **Emil Post** giới thiệu vào năm 1946. Bài toán được phát biểu như sau:

Xét hai danh sách $x = (x_1, x_2, \ldots, x_n)$ và $y = (y_1, y_2, \ldots, y_n)$ của các xâu khác rỗng trên bảng chữ $\Sigma = \{0, 1\}$. Có tồn tại hay không một dãy $i_1, i_2, \ldots, i_m$, $1 \le i_j \le n$, sao cho

$$x_{i_1} \ldots x_{i_m} = y_{i_1} \ldots y_{i_m}$$

Lưu ý: Các chỉ số i_j không nhất thiết phải khác nhau và *m* có thể lớn hơn *n*.

Ví dụ 8.12. Với hai danh sách $x = (b, bab^3, ba)$ và $y = (b^3, ba, a)$ bài toán PCP có lời giải hay không?

Giải:

Chúng ta cần xác định xem, có tồn tại hay không một dãy con của x và y tương ứng bằng nhau không. Ta có thể chọn $i_1 = 2$, $i_2 = i_3 = 1$, $i_4 = 3$, nghĩa là tồn tại dãy (2, 1, 1, 3), m = 4 để

$$\boxed{bab^3}\ \boxed{b}\ \boxed{b}\ \boxed{ba} = \boxed{ba}\ \boxed{b^3}\ \boxed{b^3}\ \boxed{a}$$

Do đó, PCP có lời giải.

Ví dụ 8.13. Chứng minh rằng với hai danh sách x =(01, 1, 1) và $y = (01^2, 10, 1^2)$ thì PCP không có lời giải.

Trước tiên chúng ta thấy, với mọi $x_i \in x$ và $y_i \in y$, ta có $|x_i| < |y_i|$, với mọi i. Do vậy, xâu được tạo ra bằng dãy các xâu con của x sẽ luôn ngắn hơn xâu được tạo ra bằng các xâu con tương ứng của y. Vì vậy, PCP không có lời giải.

Post đã phát biểu định lý sau:

Định lý 8.8 PCP trên bảng chữ Σ với $|\Sigma| \ge 2$ sẽ không quyết định được.

Sử dụng kỹ thuật rút gọn để chuyển bài toán PCP về nhiều lớp các bài toán quyết định trong ngôn ngữ hình thức. Những kết luận sau đây có thể suy ra được từ bài toán PCP [4].

(i) Nếu L_1 và L_2 hai ngôn ngữ phi ngữ cảnh trên bảng chữ Σ, $|\Sigma| \ge 2$, thì không tồn tại thuật toán để quyết định được những kết quả sau đúng hay sai:

   a. $L_1 \cap L_2 = \varnothing$,

   b. $L_1 \cap L_2$ là ngôn ngữ phi ngữ cảnh,

   c. $L_1 \subseteq L_2$,

   d. $L_1 = L_2$.

(ii) Nếu G là văn phạm cảm ngữ cảnh (loại 1), thì không tồn tại thuật toán để quyết định được những kết quả sau đúng hay sai:

a. $L(G) = \varnothing$,

b. L(G) là vô hạn,

c. $x_0 \in L(G)$ với x_0 nào đó.

(iii) Nếu G là văn phạm ngữ cấu (loại 0), thì không tồn tại thuật toán để quyết định xem $x \in \Sigma^*$ có thuộc L(G) hay không.

8.10 Lớp các bài toán NP đầy đủ

Vào đầu những năm 60 của thế kỷ 20, các máy tính đã được sử dụng rộng rãi để giải các bài toán khoa học - kỹ thuật và các bài toán kinh tế. Các nhà tin học đứng trước một vấn đề thách thức rất lớn trước những câu hỏi: thế nào là một thuật toán "tốt", hiệu quả, hay ngược lại, một thuật toán "không tốt"?, thế nào là một bài toán "dễ", một bài toán "khó"? Lúc đầu, phần lớn nhận xét theo trực quan, nhanh hay không là dựa vào kết quả thực nghiệm. Mãi cho đến 1965 Edmond lần đầu tiên đưa ý tưởng mới: *một thuật toán được gọi là hiệu quả (tốt), nếu thời gian thực hiện trên máy bị chặn lại bởi một đa thức của mọi dữ liệu vào* (kể cả trường hợp xấu nhất). Các nhà khoa học (trong đó có Steve Cook và Dick Karp) đã khẳng định rằng yêu cầu tối thiểu đối với một thuật toán hiệu quả là thời gian tính của nó phải là đa thức $\theta(n^c)$, với *c* là hằng số. Người ta cũng nhận thấy rằng đối với nhiều lớp bài toán, việc tìm ra những thuật toán như vậy là rất khó khăn, hơn nữa chúng ta còn không biết là một thuật toán như thế có tồn tại hay không. Chính vì thế, Cook và Karp cùng một số người khác đã đưa ra định nghĩa lớp các bài toán *NP-đầy đủ*, mà cho đến nay nhiều người tin rằng (nhưng chưa chứng minh được) không thể có thuật toán hiệu quả để giải chúng.

8.10.1 Các lớp bài toán P và NP

Đánh giá độ phức tạp tính toán của bài toán là xác định thời gian tính toán (số các phép toán cơ sở cần thực hiện) của thuật toán tốt nhất trong tất cả các thuật toán giải bài toán đó, cả những thuật toán đã biết lẫn các thuật toán còn chưa biết. Việc đánh giá độ phức tạp tính toán của bài toán giữ vai trò định hướng trong việc thiết kế các thuật toán để giải những bài toán đặt ra trong thực tế.

Để định nghĩa được lớp các bài toán **P** và **NP** chúng ta cần định nghĩa độ phức tạp thời gian của máy Turing.

Định nghĩa 8.7 Độ phức tạp thời gian T(n) của máy Turing (đơn định) là số phép chuyển cực đại của MT để xử lý một dãy vào bất kỳ có độ dài *n*.

Định nghĩa 8.8 Ngôn ngữ L được gọi là thuộc lớp **P** nếu tồn tại một máy Turing (đơn định) MT sao cho MT có độ phức tạp thời gian là đa thức *p(n)* và nó đoán nhận được L.

Mục 8.5 chúng ta đã định nghĩa máy Turing không đơn định. Đặc điểm của máy Turing không đơn định là tại mỗi hiện trạng bất kỳ, máy được phép có nhiều khả năng dịch chuyển (hàm chuyển không đơn trị). Để định nghĩa được lớp **NP** chúng ta cần định nghĩa độ phức tạp thời gian của máy Turing không đơn định.

Định nghĩa 8.9 Máy Turing không đơn định MT có độ phức tạp thời gian T(n) nếu với mọi dãy vào được đoán nhận có độ dài *n,* tồn tại một dãy nhiều nhất T(n) phép chuyển để MT đạt đến hiện trạng chấp nhận.

Định nghĩa 8.10 Ngôn ngữ L được gọi là thuộc lớp **NP** nếu tồn tại một máy Turing không đơn định MT sao cho MT có độ phức tạp thời gian là đa thức *p(n)* và nó đoán nhận L.

Mặc dù định nghĩa của lớp **P** và **NP** là tương tự, song chúng có sự khác biệt rất lớn.

- Khi L thuộc **P**, số các bước chuyển của MT để xử lý *dãy đầu vào bất kỳ* có độ dài *n* là nhỏ hơn hoặc bằng *p(n)*.
- Khi L thuộc **NP**, số các bước chuyển của MT để xử lý *là nhỏ hơn hoặc bằng p(n) chỉ đối với những dãy vào được chấp nhận*. Như vậy, trong lớp **NP,** giới hạn *p(n)* các phép chuyển chỉ có tác dụng khi chúng ta có thể tìm được dãy đầu vào $w \in L$.

Một cách hoàn toàn tương đương, ta có thể hiểu **P** là *lớp các bài toán có thể giải được trong thời gian đa thức*, còn **NP** là lớp các bài toán, *mà mọi nghiệm giả định đều có thể kiểm định được trong thời gian đa thức*. Thường thì việc tìm nghiệm khó hơn nhiều so với việc kiểm chứng nghiệm.

Sau đây là một số tính chất đóng của lớp **P** và **NP** [1].

Định lý 8.9 Nếu $A, B \in \mathbf{P}$ thì $A \cup B$, $A \cap B$, AB, $A^* \in \mathbf{P}$

Định lý 8.10 Nếu $A, B \in \mathbf{NP}$ thì $A \cup B$, $A \cap B$, AB, $A^* \in \mathbf{NP}$.

Một trong những bài toán "mở" quan trọng nhất hiện nay của khoa học máy tính lý thuyết là **P = NP**? Nói cách khác, có phải mọi ngôn ngữ được đoán nhận bởi một thuật toán không đơn định với thời gian đa thức thì cũng được chấp chận bởi thuật toán đơn định nào đó với thời gian đa thức hay không? Đây là một trong những bài toán khó nhất đặt ra trong thế kỷ 21 này cho các nhà toán học và khoa học tính toán.

8.10.2 NP-đầy đủ

Hiển nhiên là $\mathbf{P} \subseteq \mathbf{NP}$, nên việc xét mối quan hệ giữa **P** và **NP** nói chung là phải duyệt lại toàn bộ lớp **NP**. Thay vì phải duyệt toàn bộ lớp **NP**, ta muốn chỉ duyệt một bộ phận nhỏ, thậm chí chỉ một vài bài toán trong **NP** mà thôi. Muốn thế ta hãy chọn ra một bài toán "*khó giải nhất*" **C** theo một nghĩa nào đó bất kỳ trong **NP** rồi kiểm tra xem nó có thuộc **P** hay không. Nếu **C** không thuộc **P** thì đó đã là bằng chứng $\mathbf{P} \subset \mathbf{NP}$. Như vậy, mỗi bài toán "khó nhất" trong **NP** là chìa khoá để giải bài toán **P = NP**?. S. Cook gọi những bài toán khó nhất trong **NP** là các bài toán **NP-đầy đủ** (NP-completeness). Vấn đề đặt ra là định nghĩa như thế nào bài toán A là khó hơn bài toán B.

Để giải quyết vấn đề trên, ta có thể vận dụng khái niệm *dẫn được* trong lý thuyết thuật toán.

Định nghĩa 8.11 Giả sử L_1, L_2 là hai ngôn ngữ trên Σ_1 và Σ_2 tương ứng. Ta nói L_1 dẫn đa thức được về L_2, ký hiệu $L_1 \preccurlyeq_p L_2$ nếu

(i) Tồn tại một hàm f: $\Sigma_1^* \rightarrow \Sigma_2^*$ sao cho $w \in L_1$ khi và chỉ khi $f(w) \in L_2$.

(ii) f() có thể tính được bởi một máy Turing có độ phức tạp thời gian là đa thức.

Định nghĩa 8.12 Ngôn ngữ $L \subseteq \Sigma^*$ là **NP-đầy đủ** nếu

(i) L là **NP**,

(ii) Với mọi ngôn ngữ L' thuộc **NP,** L dẫn đa thức được về L'.

Định nghĩa của lớp các ngôn ngữ **P** và **NP** như trên có thể mở rộng cho lớp các bài toán. Bài toán **C** thuộc **NP-đầy đủ**, nếu nó thuộc **NP** và mọi bài toán **B** thuộc **NP** đều dẫn đa thức được về **C** theo nghĩa ngôn ngữ đoán nhận bởi máy Turing giải **B** dẫn đa thức được về ngôn ngữ đoán nhận bởi máy Turing giải **C**. Hiện nay có hàng trăm bài toán **NP-đầy đủ** thuộc đủ mọi lĩnh vực: tính toán mệnh đề, lý thuyết đồ thị, lý thuyết tối ưu, qui hoạch, lý thuyết tổ hợp, lập lịch, ...

Ta có các mệnh đề sau:

Mệnh đề 8.1 Nếu $L_1 \preccurlyeq_p L_2$ và $L_2 \in$ **P**, thì $L_1 \in$ **P**.

Chứng minh: dùng định nghĩa của quan hệ $\preccurlyeq_p$.

Mệnh đề 8.2 Nếu L_1 là **NP-đầy đủ**, $L_2 \in$ **NP** và $L_1 \preccurlyeq_p L_2$, thì $L_2 \in$ **NP-đầy đủ**.

Chứng minh: dùng tính bắc cầu của quan hệ $\preccurlyeq_p$.

Về ý nghĩa, mệnh đề 8.2 cho một phương pháp cơ bản để chứng minh một bài toán mới là NP-đầy đủ trên cơ sở dẫn được đa thức về từ những bài toán NP-đầy đủ cho trước.

Định lý 8.11 Giả sử L là NP-đầy đủ, **P** = **NP** khi và chỉ khi L thuộc **P**.

Chứng minh dùng mệnh đề 8.1.

Định lý 8.9 chỉ ra con đường nhằm tới đích **P = NP**.

Tuy nhiên để áp dụng được mệnh đề 8.2, ta còn cần có một ngôn ngữ đầu tiên là NP-đầy đủ. Mặt khác, những bài toán NP-đầy đủ còn có những tính chất sau [4]:

Tính chất 8.1 Không tìm được thuật toán thời gian đa thức cho bất kỳ một bài toán NP-đầy đủ nào.

Tính chất 8.2 Không thể khẳng định được là không tồn tại thuật toán thời gian đa thức đối với những bài toán NP-đầy đủ.

Tính chất 8.3 Nếu tìm được một thuật toán thời gian đa thức để giải một bài toán NP-đầy đủ thì sẽ tồn tại thuật toán thời gian đa thức cho tất cả các bài toán NP-đầy đủ.

Tính chất 8.4 Nếu chứng minh được rằng có một thuật toán thời gian đa thức cho một bài toán NP-đầy đủ thì sẽ tồn tại thuật toán thời gian đa thức cho mọi bài toán NP-đầy đủ.

Do vậy, để trả lời cho câu hỏi **P = NP**?, chúng ta cần biết đầy đủ thông tin về những bài toán NP-đầy đủ. Cook đã chứng minh SAT là NP-đầy đủ.

Định lý 8.12 (Cook Theorem) SAT là NP-đầy đủ.

Cook là người đầu tiên chứng minh được bài toán SAT là NP-đầy đủ vào năm 1971. Một năm sau đó, dựa vào phương pháp của Cook, M. Karp đã chỉ ra một loạt khoảng 20 bài toán tối ưu tổ hợp cổ điển là NP-đầy đủ. Đến năm 1979, hai tác giả M. Garey và D. Johnson đã tổng kết được khoảng 300 bài toán là NP-đầy đủ. Từ đó đến nay, con số các bài toán là NP-đầy đủ vẫn tăng liên tục hàng năm.

Ngoài những bài toán đã nêu ở trên, ta có thể nêu thêm một số bài toán trong các lĩnh vực sau đây là NP-đầy đủ [1]:

1. Bài toán hành trình của người bán hàng (Travelling Salesman Problem). Cho trước *n* thành phố, biết khoảng cách giữa chúng và một số D, có tồn tại một hành trình để người bán hàng đi qua tất cả các thành phố đó với quãng đường đi nhiều nhất là D?.
2. Bài toán qui hoạch 0-1 (Zero-one Programming Problem). Cho trước m phương trình

   $$\sum_{j=1}^{m} a_{ij}x_j = b_j,\ i = 1, 2, \ldots, n$$

   Có tồn tại các giá trị 0, 1 để thoả mãn hệ phương trình trên hay không?
3. Bài toán qui hoạch nguyên tuyến tính dạng quyết định. Cho trước ma trận nguyên $A = (a_{ij})$ kích thước $m \times n$, $b = (b_1, \ldots, b_m)$ là vector nguyên. Hỏi hệ bất phương trình $Ax \geq b$ có nghiệm nguyên hay không?
4. Bài toán tìm chu trình Hamilton của một đồ thị cho trước. Chu trình Hamilton của đồ thị G là chu trình đi qua tất cả các đỉnh đúng một lần, trừ đỉnh đầu và đỉnh cuối.
5. Bài toán tìm tập phủ các đỉnh của đồ thị. Cho trước đồ thị G = (V, E) và số nguyên $K \geq 0$, liệu V có một tập con V' có cỡ $\leq K$ (có K đỉnh) sao cho mỗi cung $e \in E$ có ít nhất một đỉnh cuối thuộc V'.
6. Bài toán tìm tập độc lập trong đồ thị. Cho trước đồ thị G = (V, E) và số nguyên $K \geq 0$, xác định xem G có tập độc lập cỡ K hay không?., nghĩa là liệu có một tập con $V' \subseteq V$, $|V'| = K$, sao cho $\forall u, v \in V', \{u, v\} \notin E$.
7. Bài toán tìm tổng con. Cho trước một dãy các số nguyên $(a_1, a_2, \ldots, a_n)$ và số nguyên S, xác định xem có tồn tại một tập $I \subseteq \{1, 2, \ldots, n\}$ sao cho $\Sigma_{i \in I} a_i = S$.

…

Sự phong phú và đa dạng của các bài toán NP-đầy đủ là một thuận lợi trong việc chọn "chìa khoá" để mở cánh cửa **P = NP**?

Trong khi chờ đợi câu trả lời bài toán quan trọng nêu trên, ta hãy thử hỏi điều gì sẽ xảy ra nếu **P = NP**, và nếu như **P ≠ NP**?

Nếu như **P ≠ NP,** các điều sau sẽ xảy ra:

- Độ mật của các mã khoá công khai sẽ được khẳng định. Do vậy, mã khoá công khai sẽ được triển khai diện rộng rãi hơn, phù hợp với xu thế phát triển các hệ thống mở, nhất là thương mại điện tử của xã hội trong tương lai.
- Những bài toán NP-đầy đủ trở thành các bài toán *bất trị*, cho đến khi có một cuộc cách mạng mới trong công nghệ thông tin cùng với một thế hệ máy tính hoàn toàn mới về nguyên lý hoạt động, có khả năng "siêu" tăng tốc tính toán. Chúng ta đã nghe nói đến những ý tưởng táo bạo của các nhà khoa học về một thế hệ máy tính mới có tên là *máy tính lượng tử*. Máy tính lượng tử sẽ hoạt động theo nguyên lý chung của cơ học lượng tử. Hy vọng máy tính lượng tử sẽ giúp chúng ta giải được những bài toán NP-đầy đủ.

Ngược lại, nếu **P = NP**, ta sẽ có những hệ quả trực tiếp sau đây:

- Mọi bài toán hễ kiểm chứng dễ thì giải cũng dễ.
- Tất cả các bài toán tối ưu, tổ hợp thông thường đều giải được trong thời gian đa thức.
- Mối lo "Bùng nổ tổ hợp" bấy lâu nay sẽ được giải quyết.
- Một loạt các hệ mã khoá công khai dựa trên giả thiết **P ≠ NP** bị đổ vỡ, ...

Nói chung người ta tin là **P ≠ NP**.

Bài tập về các bài toán quyết định được và lớp bài toán P, NP-đầy đủ

8.1 Xây dựng máy Turing để đoán nhận $\{0^n1^n \mid n \geq 1\}$

8.2 Xác định dãy tính toán đối với các dãy 1213, 2133, 312 của máy Turing cho trước ở ví dụ 8.7.

8.3 Thiết kế máy Turing để đoán nhận tất cả các xâu trên {0, 1} có chứa chẵn số 1.

8.4 Xây dựng máy Turing để đoán nhận ngôn ngữ $\{a^nb^nc^n \mid n, m \geq 1\}$

8.5 Xây dựng máy Turing để đoán nhận tất cả các Palindrrome chẵn trên tập {0, 1}

8.6 Chứng minh rằng vấn đề đệ qui của văn phạm loại 0 là không giải được.

8.7 Chứng minh rằng bài toán quyết định xem một ngôn ngữ cảm ngữ cảnh cho trước có phải là phi ngữ cảnh hay không là không giải được.

8.8 Chứng minh rằng $\{x \mid x$ là tập hợp và $x \notin x\}$ không phải là tập hợp (Đây là nghịch lý Russell).

8.9 Chứng minh rằng PCP là giải được nếu $|\Sigma| = 1$.

8.10 Giả sử $x = (x_1x_2, \dots x_n)$ và $y = (y_1y_2 \dots y_n)$ là hai xâu khác rỗng trên Σ với $|\Sigma| \geq 2$.

(i) PCP là giải được với n = 1?

(ii) PCP là giải được với n = 2?

8.11 Chứng minh rằng PCP với {(01, 011), (1, 10), (1, 11)} không có lời giải.

8.12 Chỉ ra rằng PCP với S = {(0, 10), (110, 0^3), ($0^2$1, 10)} không có lời giải.

Phụ lục: Hướng dẫn và lời giải các bài tập

Chương 1

1.6 Ta có thể chứng minh qui nạp theo |w|. Nếu w = Λ, hiển nhiên abΛ = Λab. Rõ ràng |w| = 0 là số chẵn, bước cơ sở qui nạp được khẳng định. Giả sử điều cần chứng minh đúng với |w| < n. Ta xét những dãy w có độ dài n và abw = wab. Bởi vì abw = wab nên w = abw_1 với w ∈ $\{a, b\}^*$. Khi đó, $ababw_1 = abw_1ab$ suy ra $abw_1 = w_1ab$. Theo giả thiết qui nạp $|w_1|$ là số chẵn. Từ đó suy ra $|w| = |w_1| + 2$ cũng là số chẵn.

1.9 2. $\forall x\ (\neg P(x, y) \rightarrow \exists y\ Q(y, x)) = (\forall x\ (P(x, y) \vee \exists y\ Q(y, x))$

$= \forall x\ \exists y\ (P(x, y) \vee Q(y, x))$

1.10 1. $A \equiv (\forall x\ P(x) \wedge \forall x\ Q(x)) \rightarrow \neg(\exists x\ R(x) \vee \exists x\ F(x))$

Thực hiện các phép biến đổi tương đương đối với A:

a/ Khử phép kéo theo →: $A = (\forall x\ P(x) \wedge \forall x\ Q(x)) \rightarrow \neg(\exists x\ R(x) \vee \exists x\ F(x))$

$= \neg(\forall x\ P(x) \wedge \forall x\ Q(x)) \vee \neg(\exists x\ R(x) \vee \exists x\ F(x))$

b/ $= (\exists x\ \neg P(x) \vee \exists x\ \neg Q(x)) \vee (\forall x\ \neg R(x) \wedge \forall x\ \neg F(x))$

c/ $= \exists x\ (\neg P(x) \vee \neg Q(x)) \vee \forall x\ (\neg R(x) \wedge \neg F(x))$

$= \exists x\ \forall y\ ((\neg P(x) \vee \neg Q(x)) \vee (\neg R(y) \wedge \neg F(y)))$

d/ Dạng chuẩn hội: $\exists x\ \forall y\ ((\neg P(x) \vee \neg Q(x) \vee \neg R(y)) \wedge (\neg P(x) \vee \neg Q(x) \vee \neg F(y)))$

Dạng chuẩn tuyển: $\exists x\ \forall y\ (\neg P(x) \vee \neg Q(x) \vee (\neg R(y) \wedge \neg F(y))$

2. a/ Công thức $(\exists x!)\ (x^3 = 1)$ là hằng đúng còn $(\exists x!)\ (x^2 - 3x + 2 = 0)$ là sai vì $x^2 - 3x + 2 = 0$ có hai nghiệm: $x_1 = 1$, $x_2 = 2$.

b/ $(\exists x!)\ P(x) \equiv \exists x\ P(x) \wedge \forall x\ \forall y\ ((P(x) \wedge P(y)) \rightarrow x = y)$.

1.11 1. P(x, y) = "y = 2x" trên trường số Z × Z, Z là trường số nguyên.

$(\forall x)(\exists y!)\ (P(x, y) \rightarrow ((\exists y!)(\forall x)\ P(x, y))$ là sai vì $(\forall x)(\exists y!)(y = 2x) = T$ nhưng $(\exists y!)(\forall x)(y = 2x) = F$.

$(\exists y!)(\forall x)\ (P(x, y) \rightarrow ((\forall x)(\exists y!)\ P(x, y))$ là đúng vì $(\exists y!)(\forall x)(y = 2x) = F$.

2. Trường để $(\exists x!)(x > 2)$ đúng, ví dụ $D = \{y \mid y \leq 2\} \cup \{3\}$.

Trường để $(\exists x!)(x > 2)$ sai, ví dụ D = Z tất cả các số nguyên.

3. $(\exists x\ \neg P(x) \vee (\forall x\ \neg Q(x)) \rightarrow (X \rightarrow (\forall x\ P(x) \wedge \exists x\ Q(x)))$

$\equiv (\exists x\ \neg P(x) \vee (\exists x\ Q(x)) \vee (\neg X \vee (\forall x\ P(x) \wedge \exists x\ Q(x)))$

$\equiv (\exists x\ \neg P(x) \vee \exists x\ Q(x) \vee \neg X) \vee (\forall x\ P(x) \wedge \exists x\ Q(x)))$

$\equiv (\exists x\, \neg P(x) \vee \exists x\, Q(x) \vee \neg X \vee \forall x\, P(x)) \wedge (\exists x\, \neg P(x) \vee \exists x\, Q(x) \vee \neg X \vee \exists x\, Q(x))$

$\equiv (\exists x\, \neg P(x) \vee \exists x\, Q(x) \vee \neg X \vee \forall x\, P(x)) \wedge (\exists x\, \neg P(x) \vee \exists x\, Q(x) \vee \neg X)$

$\equiv (\exists x\, \neg P(x) \vee \exists x\, Q(x) \vee \neg X)$, vì $\exists x\, \neg P(x) \vee \forall x\, P(x) = \neg(\forall x\, P(x)) \vee \forall x\, P(x) = T$

$\equiv (\forall x)(\exists y)\, ((P(x) \wedge Q(y)) \vee \neg X)$.

1.12 1. Công thức $(\exists x)(\forall x)\, P(x, y)$ trên trường $M \times M$ với $M = \{1, 2, 3\}$ ở dạng không còn các lượng tử $\exists$, $\forall$ mà chỉ còn các phép hội, tuyển và các vị từ trên trường đã cho:

$(P(1,1) \wedge P(1,2) \wedge P(1,3)) \vee (P(2,1) \wedge P(2,2) \wedge P(2,3)) \vee (P(3,1) \wedge P(3,2) \wedge P(3,3))$.

2. $A \equiv (\forall x\, P(x) \rightarrow \exists x\, Q(x)) \vee \exists x\, \neg P(x) \vee \exists x\, \neg Q(x) \vee (((X \rightarrow Y) \wedge \neg Y) \rightarrow \neg X)$.

a/ $A \equiv (\neg(\forall x\, P(x)) \vee \exists x\, Q(x)) \vee \exists x\, \neg P(x) \vee \exists x\, \neg Q(x) \vee (\neg((\neg X \vee Y) \wedge \neg Y) \vee \neg X)$

b/ $A \equiv (\exists x\, \neg P(x) \vee \exists x\, Q(x)) \vee \exists x\, \neg P(x) \vee \exists x\, \neg Q(x) \vee (((X \wedge \neg Y) \vee Y) \vee \neg X)$

c/ $A \equiv \exists x\, (\neg P(x) \vee (\neg X \wedge Y))$

d/ Dạng chuẩn hội: $\exists x\, ((\neg P(x) \vee \neg X) \wedge (\neg P(x) \vee Y))$

Dạng chuẩn tuyển: $\exists x\, (\neg P(x) \vee (\neg X \wedge Y))$.

1.13 3. Suy luận dưới đây là đúng.

$$\begin{array}{l} \neg X_1 \vee X_2 \\ X_3 \vee \neg X_2 \\ X_4 \vee \neg X_3 \\ \underline{(\neg X_4 \vee X_5) \wedge \neg X_5} \\ \quad \therefore \neg X_1 \end{array}$$

Áp dụng tính chất của qui tắc suy diễn ta có: $\neg X_1 \vee X_2 = X_1 \rightarrow X_2$ và $X_3 \vee \neg X_2 = X_2 \rightarrow X_3$. Sử dụng qui tắc suy diễn bắc cầu suy ra $X_1 \rightarrow X_3$. Tương tự $X_4 \vee \neg X_3 = X_3 \rightarrow X_4$, $\neg X_4 \vee X_5 = X_4 \rightarrow X_5$, suy ra $X_1 \rightarrow X_4$ và cuối cùng $X_1 \rightarrow X_5$. Sau đó sử dụng qui tắc Modus tollens:

$$\begin{array}{l} \neg X_5 \\ \underline{X_1 \rightarrow X_5} \\ \therefore \neg X_1 \end{array}$$

1.15 1. a/ Công thức tương đương

$F = ((\forall x)(\neg P(x) \rightarrow Q(x)) \wedge (\exists x)\, \neg P(x) \wedge (\forall x)(Q(x) \rightarrow R(x)) \wedge (\forall x)(S(x) \rightarrow \neg R(x))) \rightarrow (\exists x)\, \neg S(x)$

Trước khi tìm dạng chuẩn tuyển, ta thực hiện một số phép biến đổi và rút gọn công thức. Ta đặt $A = (\forall x)(\neg P(x) \rightarrow Q(x)) \wedge (\forall x)(Q(x) \rightarrow R(x)) \wedge (\forall x)(S(x) \rightarrow \neg R(x))$

$= (\forall x)((\neg P(x) \rightarrow Q(x)) \wedge (Q(x) \rightarrow R(x)) \wedge (S(x) \rightarrow \neg R(x)))$.

Dễ nhận thấy: $S(x) \to \neg R(x) = R(x) \to \neg S(x)$. Sử dụng qui tắc suy diễn bắc cầu hai lần suy ra $A = (\forall x)((\neg P(x) \to \neg S(x)$. Do vậy,

$$F = ((\forall x)((\neg P(x) \to \neg S(x) \wedge (\exists x)\neg P(x)) \to (\exists x)\neg S(x)$$

$$= (\neg((\forall x)((P(x) \vee \neg S(x)) \vee \neg(\exists x)\neg P(x)) \vee (\exists x)\neg S(x)$$

$$= ((\exists x)((\neg P(x) \wedge S(x)) \vee \neg((\exists x)\neg P(x)) \vee (\exists x)\neg S(x)$$

$$= ((\exists x)((\neg P(x) \wedge S(x)) \vee (\exists x)\neg S(x) \vee \neg((\exists x)\neg P(x))$$

$$= ((\exists x)((\neg P(x) \wedge S(x) \vee \neg S(x)) \vee \neg((\exists x)\neg P(x))$$

$$= (\exists x)\neg P(x) \vee (\exists x)\neg S(x)) \vee \neg((\exists x)\neg P(x)) = T$$

b/ Mô hình suy diễn trên có đúng trên trường M. Tương tự như trên, sử dụng quy tắc bắc cầu và qui tắc Modus Ponens để khẳng định điều đó.

2. Hãy diễn đạt định nghĩa giới hạn $\underset{x \to x_0}{Limf}(x) = L$ dưới dạng một công thức vị từ:

$$\forall \varepsilon\, \exists \delta\ \forall x\, (0 < |x - x_0| < \delta \to |f(x) - f(x_0)| < \varepsilon)$$

1.16 1. Đặt P(x) = "X có bằng cử nhân" và Q(x) = "X đã tốt nghiệp đại học". Đặt tiếp R = "Lan". Từ phát biểu của bài toán ta có:

$(\forall X\, (P(X) \to Q(X)) \wedge P(R)) \to Q(R)$ là hằng đúng.. Nghĩa là những phát biểu trên là đúng.

Chương 2

2.2 Bởi vì $\sigma(q_1,a) = \sigma(q_1,a)$, nên R là phản xạ. Tính đối xứng là hiển nhiên. Nếu q_1Rq_2 thì $\sigma(q_1,a) = \sigma(q_2,a)$ với $a \in \Sigma$, còn nếu q_2Rq_3 thì $\sigma(q_1,a) = \sigma(q_2,a)$. Từ đó suy ra $\sigma(q_1,a) = \sigma(q_3,a)$, nghĩa là q_1Rq_3. Vậy R là quan hệ tương đương.

2.3 Bảng chuyển trạng thái của ÔTKĐĐ để đoán nhận {ab, ba}

Bảng A.1

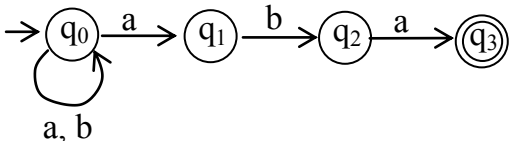
Trạng thái/Σ	a	b
→ q_0	q_1	q_2
q_1		q_3
q_2	q_1	
q_3		

Bảng chuyển trạng thái của ÔTĐĐ tương đương với bảng A.2 là

Bảng A.2

Trạng thái/∑	a	b
→ $[q_0]$	$[q_1]$	$[q_2]$
$[q_1]$	∅	$[q_3]$
$[q_2]$	$[q_1]$	∅
$[q_3]$	∅	∅
∅	∅	∅

2.5 Ôtômát không đơn định đoán nhận được tập tất cả các xâu từ tập {a,b} kết thúc bằng xâu con aba được xây dựng như sau



2.6 Bảng chuyển trạng thái của ôtômát đơn định tương đương là

Bảng A.3

Trạng thái/∑	0	1	2
→ $[q_0]$	$[q_1, q_4]$	$[q_4]$	$[q_2, q_3]$
$[q_4]$	∅	∅	∅
$[q_1, q_4]$	∅	$[q_4]$	∅
$[q_2, q_3]$	∅	$[q_4]$	$[q_2, q_3]$
∅	∅	∅	∅

2.9 Cho trước $\Sigma = \{a_1, a_2, \ldots, a_n\}$, tìm ôtômát hữu hạn để đoán nhận ngôn ngữ

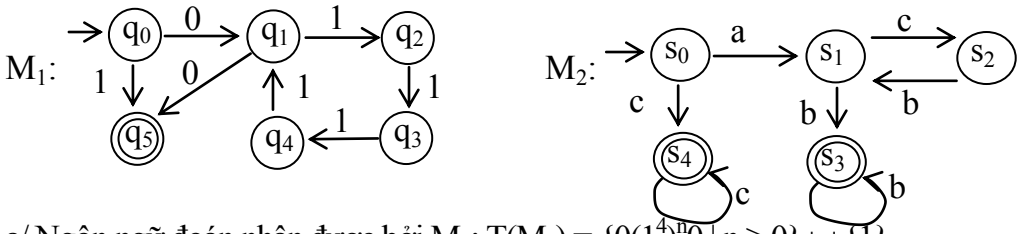
a) $L_1 = \{x \in \Sigma^* \mid x$ có độ dài lẻ$\}$ được đoán nhận bởi ôtômát có hàm chuyển trạng thái như sau

$Q \backslash \Sigma$	a_1	a_2	a_3	. . .	a_{n-1}	a_n
→ q_0	q_1	q_1	q_1	. . .	q_1	q_1
q_1	q_0	q_0	q_0	. . .	q_0	q_0

c) $L_1 = \{x \in \Sigma^* \mid x$ có độ dài chẵn và lớn hơn 1$\}$ được đoán nhận bởi ôtômát có hàm chuyển trạng thái như sau

$Q \backslash \Sigma$	a_1	a_2	a_3	. . .	a_{n-1}	a_n
→ q_0	q_1	q_1	q_1	. . .	q_1	q_1
q_1	q_2	q_2	q_2	. . .	q_2	q_2
q_2	q_1	q_1	q_1	. . .	q_1	q_1

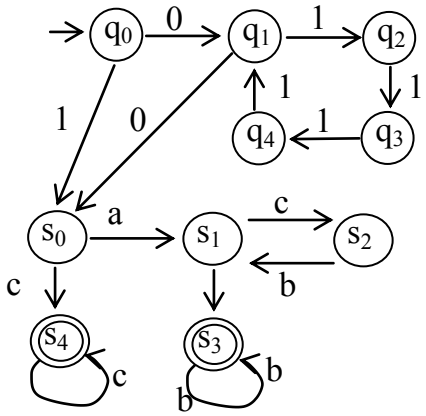
2.10* Cho hai ôtômát sau đây:



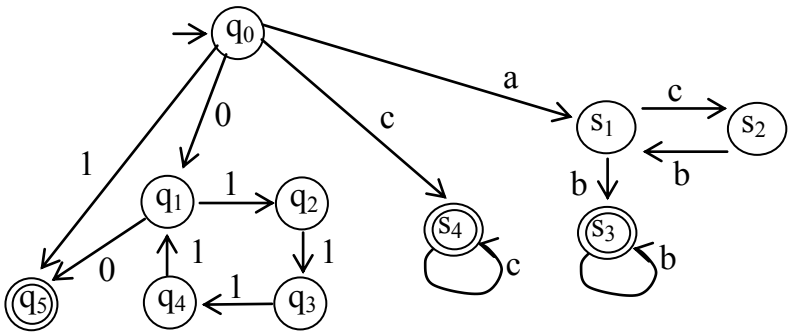
a/ Ngôn ngữ đoán nhận được bởi M_1: $T(M_1) = \{0(1^4)^n0 \mid n \geq 0\} \cup \{1\}$

Ngôn ngữ đoán nhận được bởi M_2: $T(M_2) = \{a(cb)^nb^m \mid n \geq 0, m \geq 1\} \cup \{c^m \mid m \geq 1\}$

b/ Ôtômát M được ghép từ M_1 và M_2 sao cho $T(M) = T(M_1)T(M_2)$, có thể xây dựng như sau:



c/ M là hợp của M_1 và M_2 sao cho $T(M) = T(M_1) \cup T(M_2)$



Chương 3

3.1 (a) $S \overset{*}{\underset{G}{\Rightarrow}} 0^nS1^n \overset{*}{\underset{G}{\Rightarrow}} 0^nA1^m1^n$, $n \geq 0$, $m \geq 1$. $A \overset{*}{\underset{G}{\Rightarrow}} 1^kA \Rightarrow 1^{k+1}$, $k \geq 0$.

$0^m1^n \in L(G)$ với $n > m \geq 1$. Vậy $L(G) = \{0^m1^n \mid n > m \geq 1\}$

(d) $L(G) = \{x \in \{0, 1\}^+ \mid$ x không chứa hai số 0 liên tiếp$\}$.

3.2 (a) $G = (\{S, A, B\}, \{0, 1\}, P, S)$, trong đó P có các qui tắc $S \to 0B \mid 1A$, $A \to 0 \mid 0S \mid 1AA$, $B \to 1 \mid 1S \mid 0BB$. Chứng minh các tính chất sau qui nạp theo $|w|$.

(i) $S \overset{*}{\Rightarrow} w$ khi và chỉ khi w chứa số các số 0 bằng số các số 1.

(ii) $A \overset{*}{\Rightarrow} w$ khi và chỉ khi w chứa số các số 0 lớn hơn số các số 1 là 1.

(iii) $B \overset{*}{\Rightarrow} w$ khi và chỉ khi w chứa số các số 1 lớn hơn số các số 0 là 1.

(b) G ó các qui tắc suy dẫn $S \to 0S1 \mid 0A1$, $A \to 1A0 \mid 10$. Hiển nhiên $L(G) \subseteq \{0^n1^m0^m1^n \mid m, n \geq 1\}$. Ngược lại áp dụng qui tắc $S \to 0S1$ (n-1) lần, $S \to 0A1$ một lần và $A \to 1A0$ (m-1) lần rồi $A \to 10$ là chúng ta thu được $\{0^n1^m0^m1^n\}$. Nghĩa là $\{0^n1^m0^m1^n \mid m, n \geq 1\} \subseteq L(G)$.

(c) G có các dẫn xuất $S \to 0S11 \mid 011$.

3.6 (a) Văn phạm cảm ngữ cảnh nhưng không phải là phi ngữ cảnh, ví dụ văn phạm có có các dẫn xuất: $S \to aS \mid b$, $aS \to aa$, $B \to a$.

(b) Văn phạm phi ngữ cảnh nhưng không phải là chính qui, ví dụ văn phạm có có các dẫn xuất: $S \to AS \mid a$, $A \to a$.

3.7 (a) $G = (\{S, A, B\}, \{0, 1\}, P, S)$, trong đó P có các dẫn xuất $S \to 0S1 \mid 0A$, $S \to 1B \mid 0 \mid 1$, $A \to 0A \mid 0$, $B \to 1B \mid 1$. Sử dụng $S \to 0S1$, $S \to 0$, hoặc $S \to 1$thì nhận được 0^m1^n, trong đó m và n khác nhau là 1. Để nhận được số các chữ số 0 nhiều hơn số các chư số 1 ta áp dụng $A \to 0A \mid 0$, $S \to 0A$ lặp lại nhiều lần, ngược lại để nhận được số các số 1 nhiều hơn thì áp dụng $S \to 1B$, $B \to 1B \mid 1$ lặp lại nhiều lần.

(c) G có các dẫn xuất $S \to aSc \mid ac \mid bc \mid bS_1c$, $S_1 \to bS_1c \mid bc$. Một xâu của L dạng $a^lb^mc^n$, $l + m = n$, có thể viết thành $a^lb^mc^mc^l$ với $l, m \geq 0$. Áp dụng l-lần qui tắc $S \to aSc$, thu được a^lSc^l. Nếu m = thì áp dụng tiếp $S \to ac$, ngược lại áp dụng $S \to bS_1c$ rồi sử dụng (m-1) lần qui tắc $S_1 \to bS_1c$, cuối cùng là $S_1 \to bc$.

3.8 (a) G có các dẫn xuất $S \to aS_1 \mid aS_2$, $S_1 \to aS$, $S_2 \to a$

(b) G có các dẫn xuất $S \to aS \mid bS \mid a$

(c) G có các dẫn xuất $S \to aS_1$, $S_1 \to aS_1 \mid bS_1 \mid a \mid b$

3.9 Chỉ ra rằng $L(G_1) = \{a^nb^n \mid n \geq 1\} = L(G_2)$.

3.11 Giả sử G_1 là văn phạm gồm các qui tắc $A \to xB \mid y$, trong đó $x, y \in \Sigma^*$ và A, B. Thay $A \to xB$, trong đó $x = a_1a_2\ldots a_n$ bằng $A \to a_1A_1$, $A_1 \to a_2A_2$, …, $A_{n-1} \to a_nB$. Những dẫn xuất $A \to y$, $y = b_1b_2\ldots b_n$ được thay bằng $A \to b_1B_1$, $B_1 \to b_2B_2$, …, $b_{n-1} \to b_n$. Dễ dàng chứng minh rằng văn phạm mới G_2 gồm những dẫn xuất mới là chính qui và tương đương với G_1.

3.12 (i) Văn phạm phi ngữ cảnh sinh ra $L = \{a^nb^n0, c^m1^kd^m \mid n > 0, m > 0, k \geq 0\}$ gồm những qui tắc: $S \to S_10 \mid cSd \mid cS_2d \mid cd$, $S_1 \to aS_1b \mid ab$, $S_2 \to 1S_2 \mid 1$.

3.13 (i) Văn phạm phi ngữ cảnh sinh ra $L = \{a^n0^kb^nc^md^m \mid n > 0, m > 0, k \geq 0\}$ gồm những qui tắc: $S \to S_1S_2$, $S_1 \to aS_1b \mid ab \mid aS_3b$, $S_3 \to 0S_3 \mid 0$, $S_2 \to cS_2d \mid cd$.

3.16 G có các qui tắc $S \to 0AB$, $A0 \to S0B$, $A1 \to SB1$, $B \to SA \mid 01$. Dễ nhận ra rằng $S \Rightarrow 0AB \Rightarrow 0ASA \Rightarrow 0A0ABA$ hoặc $S \Rightarrow 0AB \Rightarrow 0A01 \Rightarrow 0S0B1$. Khi áp dụng $S \to 0AB$ thì nhận được A trong các dẫn xuất. Nếu muốn loại bỏ A thì phải áp dụng $A0 \to S0B$, $A1 \to SB1$, khi đó S lại xuất hiện ở vế phải. Do vậy một trong hai biến S, A không thể loại bỏ được ở vế phải, nghĩa là $L(G) = \varnothing$.

3.17 (a) Sai. Nếu G_1 gồm các dẫn xuất $S \to SS \mid aS \mid bS \mid a \mid b$ là phi ngữ cảnh, và G_2 có các qui tắc $S \to aS \mid bS \mid a \mid b$ là chính qui. Hai văn phạm này tương đương với nhau vì $L(G_1) = L(G_2) = \{a, b\}^+$.

(b) Đúng. $L = \{w_1, w_2, \ldots, w_n\}$ thì văn phạm có những qui tắc sau: $S \to w_1 \mid w_2 \mid \ldots \mid w_n$ là ngôn ngữ phi ngữ cảnh.

(c) Đúng. Chỉ cần chỉ ra rằng $w \in \Sigma^*$ tồn tại văn phạm chính qui để sinh ra w.

3.18 $G = (\{S, S_1, A, B\}, \{0, 1, \ldots, 9\}, P, S)$. P gồm có $S \to 0 \mid 2 \mid 4 \mid 6 \mid 8 \mid AS_1$, $S_1 \to 0 \mid 2 \mid 4 \mid 6 \mid 8$, $S \to ABS_1$, $B \to 1 \mid 2 \mid \ldots \mid 9$, $A \to 1 \mid 2 \mid \ldots \mid 9$.

3.20 Tính đối xứng của cụm từ trong L thể hiện ở chỗ: trong mỗi xâu thì số vị trí ký hiệu a luôn luôn nhiều hơn số vị trí của b. Do đó khi sinh ra một vị trí của b thì cũng phải sinh ra một ví trí a và phải sinh thêm ít nhất một vị của a nữa. Từ đó văn phạm sinh ra L có các qui tắc sau:

$S \to aSb \mid bSa \mid aAb \mid bAa$, $A \to aA \mid a$

3.24 Vì $m \geq n \geq 1$ nên có thể viết $m = n + k$, $k \geq 0$. Khi đó

a/ $L = \{a^{n+k}b^n \mid 0 \leq k, 1 \leq n\} = \{a^na^kb^n \mid 0 \leq k, 1 \leq n\}$

Văn phạm G sinh ra L do vậy có các qui tắc

$S \to aSb \mid aAb \mid ab$, $A \to aA \mid a$

Đây là văn phạm phi ngữ cảnh.

b/ Dạng chuẩn Chomsky của G:

$S \to DB \mid DC \mid DM$, $D \to a$, $B \to SM$, $M \to b$, $C \to AM$, $A \to DA \mid a$

Chương 4

4.1 (i) **0 + 1 + 2**

(ii) $\mathbf{b^*ab^*}$

(iii) $\mathbf{aa(aaa)^*}$

(iv) $\mathbf{(aa)^* + (aaa)^* + aaaaa}$

4.2 (i) Đó là các xâu:

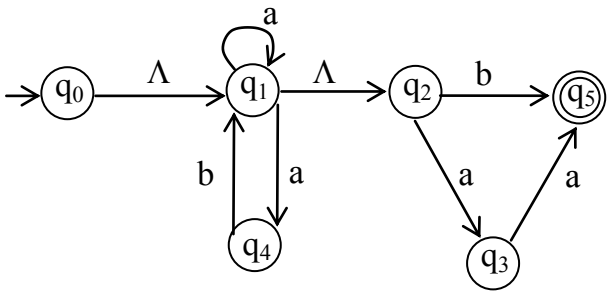
+ Được ghép từ hai xâu: xâu có độ dài nhiều nhất là 3 của $(ab+a)^*$ và aa

+ Được ghép từ hai xâu: xâu có độ dài nhiều nhất là 4 của $(ab+a)^*$ và b.

Đó là các xâu: aa, aaa, abaa, aaaa, aabaa, abaaa, a^5, ababb, aaabbb, và a^4b.

(ii) Tương tự như trên, những xâu có độ dài nhỏ hơn hoặc 5 của $\mathbf{a}^* + (\mathbf{ab} + \mathbf{a})^*$ là: Λ, a, ab, aa, abab, aab, aba, a^3, abaa, aaab, ababa, a^4, abaaa, và a^5.

4.5 Hệ chuyển đổi (hoặc ôtômát hữu hạn) tương đương với các biểu thức $(\mathbf{ab} + \mathbf{a})^*(\mathbf{aa} + \mathbf{b})$ được cho bởi hình sau.



Sau đó tiếp tục loại bỏ các cung Λ sẽ thu được ôtômát hữu hạn yêu cầu.

4.6 (a) $(\mathbf{1} + \mathbf{01})^*$ biểu diễn cho tập những xâu không chứa cặp 00. Biểu thức $(\mathbf{1} + \mathbf{01})^*\mathbf{00}(\mathbf{1} + \mathbf{01})^*$ biểu diễn cho tập những xâu chứa đúng một cặp 00. Tương tự đối với cặp 11. Vậy biểu thức chính qui biểu diễn tập tất cả các xâu trên {0, 1} có nhiều nhất một cặp số 0 hoặc nhiều nhất một cặp số 1 là:

$$(\mathbf{1} + \mathbf{01})^* + (\mathbf{1} + \mathbf{01})^*\mathbf{00}(\mathbf{1} + \mathbf{01})^* + (\mathbf{0} + \mathbf{10})^*(\mathbf{0} + \mathbf{10})^*\mathbf{11}(\mathbf{0} + \mathbf{10})^*$$

(c) $\mathbf{b}^* + (\mathbf{b} + \mathbf{abb})^*\,\mathbf{ab}^*$

(d) $\mathbf{1}(\mathbf{0} + \mathbf{1})^*\mathbf{00}$.

4.7 Giả sử L là tập tất cả các xâu *Palindrome* trên {a, b} được đoán nhận bởi ôtômát hữu hạn $M = (Q, \Sigma, \delta, q_0, F)$. Bởi vì Q là hữu hạn nên $\{\delta(q_0, a^n) \mid n \geq 1\}$ là tập con của Q, nghĩa là $\delta(q_0, a^n) = \delta(q_0, a^m)$, với $m < n$. Mặt khác, $a^nb^{2n}a^n \in L$ (là Palindrome) nên $\delta(q_0, a^nb^{2n}a^n) = q \in F$. Vì $\delta(q_0, a^n) = \delta(q_0, a^m)$ nên từ tính chất của hàm chuyển ta có $\delta(q_0, a^mb^{2n}a^n) = \delta(q_0, a^nb^{2n}a^n)$, $m < n$. Từ đó suy ra $a^mb^{2n}a^n \in L$. Điều này mâu thuẫn vì $m < n$ nên $a^mb^{2n}a^n$ không phải là Palindrome.

4.10 Giả sử $L = \{ww \mid w \in \{a, b\}^*\}$ là tập chính qui, được đoán nhận bởi ôtômát M có n trạng thái. Dễ nhận thấy số ký hiệu a, b trong các xâu của L luôn là số chẵn. Xét $w = a^nb$, hiển nhiên $a^nba^nb \in L$. $|ww| = 2(n + 1) > n$. Áp dụng bổ đề Bơm để viết $ww = xyz$ với $|y| \neq 0$, $|xy| \leq n$. Hãy tìm i sao cho $xy^iz \notin L$ thì sẽ dẫn đến mâu thuẫn.

Thật vậy, ta có thể chọn xâu y chỉ có một trong các dạng sau:

(i) y không chứa b, $y = a^k$, $k \geq 1$.

(ii) y chỉ có một ký hiệu b.

Biết rằng $|y| \leq |xy| \leq n$. Trường hợp (i), ta chọn i = 0. Khi đó $xy^0z = a^{n-k}ba^nb$ không thuộc L vì không thể viết xz thành dạng uu với $u \in \{a, b\}^*$. Tương tự trường hợp (ii) cũng chọn i = 0. Dĩ nhiên $xy^0z = xz \notin L$ vì số ký hiệu b trong xz là lẻ, mâu thuẫn.

4.11 (i) Áp dụng bổ đề Bơm để chỉ ra $L = \{a^n b^{2n} \mid n > 0\}$ không phải là chính qui. Giả sử nó là chính qui, được đoán nhận bởi ôtômát M có n trạng thái. Xét $w = a^n b^{2n}$. Theo bổ đề Bơm, $w = xyz$ với $|y| > 0$, $|xy| \le n$ và $xz \in L$. Vì $|xy| = m \le n$ nên $xy = a^m$ và $y = a^k$ với $0 < k \le n$. Vậy $xz = a^{n-k}b^{2n} \in L$ là mâu thuẫn vì $n - k \ne n$. Vậy, L không phải là chính qui.

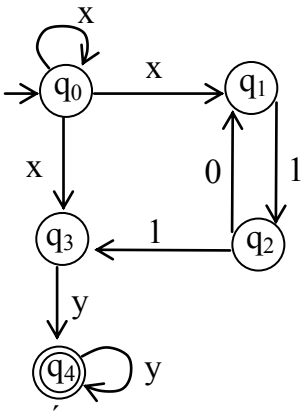
(iii) Áp dụng bổ đề Bơm để chỉ ra $L = \{a^n b^n \mid n > 0\}$ không phải là chính qui. Giả sử nó là chính qui, được đoán nhận bởi ôtômát $M = (Q, \Sigma, \delta, q_0, F)$ có n trạng thái. Bởi vì Q là hữu hạn nên $\{\delta(q_0, a^n) \mid n \ge 0\}$ là tập con của Q, nghĩa là $\delta(q_0, a^n) = \delta(q_0, a^m)$, với $m \ne n$. Do đó

$$\delta(q_0, a^m b^n) = \delta(\delta(q_0, a^m), b^n) = \delta(\delta(q_0, a^n), b^n) = q \in F$$

Suy ra $a^m b^n \in L$ với $m \ne n$ là mâu thuẫn.

4.19 (i) Văn phạm chính qui G sinh ra $L = \{x^n(10)^m y^k \mid n, k \ge 1, m \ge 0\}$ có các qui tắc sau: $S \to xS \mid xA \mid xB$, $A \to 1C$, $C \to 0A \mid 0B$, $B \to yB \mid y$.

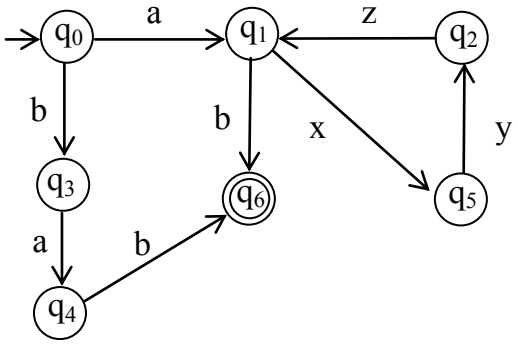
(ii) Ôtômát hữu hạn trạng thái (không đơn định) M sao cho T(M) = L(G) được biểu diễn như sau:



(iii) Dãy các qui tắc $S \Rightarrow xS \Rightarrow xxA \Rightarrow xx1C \Rightarrow xx10B \Rightarrow xx10y$, vì vậy $xx10y \in L(G)$. Xâu x101 không được sinh ra bởi văn phạm G vì không có cách nào để dẫn ra được x101 ở trạng thái kết thúc.

4.20 (i) Ngôn ngữ $L = \{a(xyz)^n b \mid n \ge 0\} \cup \{bab\}$. Sau đó chứng minh L = L(G).

(ii) Ôtômát hữu hạn trạng thái (không đơn định) M sao cho T(M) = L(G) là



(iii) Dễ dàng chỉ ra được dãy các dẫn xuất để xâu $w = axyzb \in L(G)$ và chỉ ra cây dẫn xuất trên đồ thị biểu diễn ôtômát ở hình trên đến trạng thái kết thúc q_6 và dãy nhãn trên nó là w.

4.21 (i) Văn phạm chính qui G để L(G) = L(M) có các qui tắc sau:

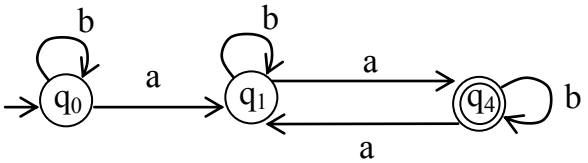
$S \to 1A \mid 0A,\ A \to xB \mid yC,\ B \to yA,\ C \to x.$

(ii) G sinh ra $w_1 = 1xyyx$, $w_2 = 0yx$ bằng cách chỉ ra hai dãy dẫn xuất tương ứng và M cũng đoán nhận được w_1, w_2 bằng cách chỉ ra hai cây dẫn xuất tương ứng.

4.22 (i) Văn phạm chính qui G để sinh ra $L = \{a^n cb^m c,\ a^n cb^m a \mid n, m \geq 0\}$ có các qui tắc sau: $S \to aS \mid cA,\ A \to bA \mid a \mid c$.

(ii), (iii) tương tự như 4.22 (ii) và (iii).

4.23 (i) Ôtômát hữu hạn trạng thái M trên {a, b} để đoán nhận tất cả các xâu có các chữ a là số chẵn và lớn hơn 0 được biểu diễn như sau:



(ii) Văn phạm chính qui tương đương với M có các qui tắc:

$S \to bS \mid aA,\ A \to bA \mid aC,\ C \to aA \mid b$

(iii) Biểu thức chính qui tương đương với ngôn ngữ sinh bởi G là

$b^*ab^*(aa)(aa)^*b^*$.

Chương V

5.2 Áp dụng các qui tắc $S \to 0S0 \mid 1S1 \mid A$ sẽ được $S \overset{*}{\Rightarrow} wAw$, với $w \in \{0, 1\}^*$. Sử dụng hai qui tắc $A \to 2B3$, $B \to 2B3$ sẽ dẫn tiếp ra $A \overset{*}{\Rightarrow} 2^m B 3^m$, sau đó áp dụng $B \to 3$ ta nhận được $S \overset{*}{\Rightarrow} w2^m3^{m+1}w$. Từ đó suy ra $L(G) \subseteq \{w2^m3^{m+1}w \mid w \in \{0, 1\}^*, m \geq 1\}$. Chiều ngược lại chứng minh tương tự.

5.4 Xét w = abababa chỉ ra rằng nó có hai cây dẫn xuất khác nhau.

5.5 Xâu abab có hai dẫn xuất khác nhau: $S \Rightarrow abSb \Rightarrow abab$ và $S \Rightarrow aAb$

$\Rightarrow abSb \Rightarrow abab$.

5.6 (i) $S \Rightarrow aB \Rightarrow aaBB \Rightarrow aaaBBB \Rightarrow aaabBB \Rightarrow aaabbB \Rightarrow aaabbaBB \Rightarrow aaabbabB \Rightarrow aaabbabbS \Rightarrow aaabbabbbA \Rightarrow aaabbabbba$.

(ii) $S \Rightarrow aB \Rightarrow aaBB \Rightarrow aaBbS \Rightarrow aaBbbA \Rightarrow aaBbba \Rightarrow aaaBbba \Rightarrow aaabBbba \Rightarrow aaabbSbba \Rightarrow aaabbaBbba \Rightarrow aaabbabbba$.

5.7 (ii) Thực hiện theo hai bước

Bước 1. Áp dụng định lý 5.3, ta có $W_1 = \{B\}$, $W_2 = \{B\} \cup \{C, A\}$, $W_3 = \{A, B, C\} \cup \{S\} = V_N$. Vậy $G_1 = G$.

Bước 2. Áp dụng định lý 5.4, sẽ nhận được $W_1 = \{S\}$, $W_2 = \{S\} \cup \{A, a\}$, $W_3 = \{S, A, a\} \cup \{B, b\}$, $W_4 = \{S, A, B, a, b\} \cup \varnothing$. Vậy, văn phạm tối giản tương đương sẽ là $G_2 = (\{S, A, B\}, \{a, b\}, P, S)$, trong đó P gồm có $S \to aAa$, $A \to bBB$, $B \to ab$.

(iii) Văn phạm tối giản tương đương sẽ là $G = (\{S, A, B\}, \{a, b\}, P, S)$, trong đó P gồm có $S \to AB$, $A \to a$, $B \to a \mid b$.

5.8 (ii) Thực hiện theo hai bước

Bước 1. $W_1 = \{C\}$ bởi vì $C \to abb$, $W_2 = \{C\} \cup \{E, A\}$ vì $E \to aC$, $A \to bCC$ $W_3 = \{A, E, C\} \cup \{S\}$ vì $S \to aAa$. $W_4 = W_3$. Vậy $G_1 = (\{S, A, C, E\}, P_1, S)$, trong đó $P_1 = \{S \to aAa, A \to Sb \mid bCC, C \to abb, E \to aC\}$.

Bước 2. Áp dụng định lý 5.4 đối với G_1, sẽ nhận được $W_1 = \{S\}$, $W_2 = \{S\} \cup \{A, a\}$, $W_3 = \{S, A, a\} \cup \{S, C, b\} = \{S, A, C, a, b\}$, $W_4 = W_3$. Văn phạm tối giản tương đương sẽ là $G_2 = (\{S, A, C\}, \{a, b\}, P_2, S)$, trong đó P_2 gồm có $S \to aAa$, $A \to Sb \mid bCC$, $c \to abb$.

5.9 (ii) Văn phạm CNF $G = (\{S, A, B, C_0, C_1, D_1, D_2), P, S)$ tương đương với văn phạm cho trước, trong đó P gồm có $S \to C_1A \mid C_0B$, $A \to 0 \mid C_0S \mid C_1D_1$, $B \to 1 \mid C_1S \mid C_0D_2$, $C_1 \to 1$, $C_0 \to 0$, $D_1 \to AA$, $D_2 \to BB$.

5.10 Văn phạm không có qui tắc rỗng và qui tắc đơn vị, do đó ta có thể thực hiện từ bước 2.

+ Thay $S \to aAD$ bằng $S \to C_aAD$ và $C_a \to a$

+ Thay $A \to aB$ bằng $A \to C_aB$

+ Thay $A \to bAB$ bằng $A \to C_bAB$ và $C_b \to b$.

Bước 3 giới hạn số các biến bên phải bằng cách

+ Thay $S \to C_aAD$ bằng $S \to C_aC_1$ và $C_1 \to AD$

+ Thay $A \to C_bAB$ bằng $A \to C_bC_2$ và $C_2 \to AB$.

Văn phạm dạng chuẩn Chomsky tương đương sẽ là:

$G = (\{S, A, B, D, C_a, C_b, C_1, C_2\}, \{a, b, d\}, P, S)$, trong đó P có các qui tắc sau: $S \to C_aC_1$, $C_1 \to AD$, $C_a \to a$, $C_b \to b$, $C_2 \to AB$, $A \to C_aB \mid C_bC_2$, $B \to b$, $D \to d$.

5.12 (i) *Bước 1*: Đặt lại tên các biến S bằng A_1, xây dựng văn phạm tương đương

$G_1 = (\{A_1, A_2, A_3\}, \{0, 1\}, P_1, A_1)$, trong đó

$P_1 = \{A_1 \to A_1A_1 \mid A_2A_1A_2 \mid A_2A_3, A_2 \to 0, A_3 \to 1$.

Bước 2: Áp dụng bổ đề 5.2 đối với $A_1 \to A_1A_1$, ta nhận được biến mới Z_1 và các qui tắc mới sẽ là: $A_1 \to A_2A_3Z_1 \mid A_2A_1A_2Z_1$, $Z_1 \to A_1 \mid A_1Z_1$.

Bước 3: Áp dụng bổ đề 5.1 đối với các A_1-dẫn xuất, ta thu được:

$A_1 \to 0A_3Z_1 \mid 0_2A_1A_2Z_1 \mid 0A_1A_3 \mid 0A_3$

Bước 4: Áp dụng bổ đề 5.1 đối với $Z_1 \to A_1$ và $Z_1 \to A_1Z_1$, thu được:

$Z_1 \to 0A_3Z_1 \mid 0_2A_1A_2Z_1 \mid 0A_1A_3 \mid 0A_3$

$Z_1 \to 0A_3Z_1Z_1 \mid 0_2A_1A_2Z_1Z_1 \mid 0A_1A_3Z_1 \mid 0A_3Z_1$

Bước 5: Văn phạm GNF tương đương với văn phạm trên có các qui tắc:

$A_1 \to 0A_3Z_1 \mid 0_2A_1A_2Z_1 \mid 0A_1A_3 \mid 0A_3$

$A_2 \to 0,\ A_3 \to 1$

$Z_1 \to 0A_3Z_1 \mid 0_2A_1A_2Z_1 \mid 0A_1A_3 \mid 0A_3$

$Z_1 \to 0A_3Z_1Z_1 \mid 0_2A_1A_2Z_1Z_1 \mid 0A_1A_3Z_1 \mid 0A_3Z_1$

5.13 (i) Giả sử $w \in L(G)$, G là CNF và $|w| = k$. Ở dạng CNF, mỗi qui tắc chỉ dẫn ra hai ký hiệu không kết thúc hoặc một ký hiệu kết thúc. Để sinh ra xâu w có độ dài k thì trước tiên phải áp dụng (k-1) lần qui tắc dạng $A \to BC$, sau đó áp dụng k-lần qui tắc dạng $A \to a$. Vậy tổng số bước thực hiện là $2k - 1$.

(ii) Khi G là GNF thì số bước dẫn xuất ra w là k ($k = |w|$) bởi vì sau mỗi lần áp dụng một qui tắc thì số ký hiệu kết thúc tăng lên 1.

5.14 (iii) Chọn $z = a^nb^nc^{2n}$. Theo bổ đề Bơm ta viết được $z = uvwxy$, sao cho $1 \le |vx| \le n$. Vậy vx không thể chứa cả ba ký hiệu a, b, c. Nếu vx chỉ chứa a, b thì chọn i sao cho uv^iwx^iy có nhiều hơn 2n lần xuất hiện của a hoặc b còn c vẫn xuất hiện 2n lần. Vậy $uv^iwx^iy \notin L$, dẫn đến mâu thuẫn. Các trường hợp còn lại cũng tương tự.

(iv) Chọn $z = a^{n^2}$, $|z| = n^2$. Theo bổ đề Bơm ta viết được $z = uvwxy$, sao cho $1 \le |vx| \le n$ (vì $|vwx| \le n$). Đặt $|vx| = m$, $m \le n$. Khi đó $|uv^2wx^2y| > n^2$ và theo bổ đề Bơm $uv^2wx^2y \in L$, nghĩa là tồn tại k sao cho $|uv^2wx^2y| = k^2$. Mặt khác $|uv^2wx^2y| = n^2 + m < n^2 + 2n + 1$, suy ra $n^2 < k^2 < (n+1)^2$, với k, n là không thể có.

5.15 *Điều kiện cần*: L = L(G), G là chính qui. Với mọi biến A trong G, $A \overset{*}{\Rightarrow} \alpha$, suy ra $\alpha = uB$, với $u \in \Sigma^*$ và $B \in V_N$. Vậy, G không phải là văn phạm tự nhúng.

Điều kiện đủ: giả sử G là văn phạm phi ngữ cảnh không tự nhúng. Xây dựng văn phạm G_1 ở dạng GNF tương đương với G. Dễ dàng nhận thấy G_1 là không tự nhúng. Đặt $|\Sigma| = n$ và m là số ký hiệu cực đại của các vế phải trong các qui tắc trong G_1. Giả sử $\alpha \in V_N^+$. Quá trình áp dụng những dẫn xuất trái nhất ta có thể chỉ ra số các biến trong α là $\le nm$. Ta định nghĩa

$G_2 = (V_N', \Sigma, P_2, S_2)$, trong đó

$V_N' = \{[\alpha] \mid |\alpha| \le nm \text{ và } \alpha \in V_N^+\}$

$S_2 = [S_1]$

$P_2 = \{[A\beta] \to b[\alpha\beta] \mid A \to b\alpha \in P_1, \beta \in V_N' \text{ và } |\alpha\beta| \le nm$

Dễ dàng nhận thấy G_2 là chính qui và $L(G_2) = L(G_1) = L(G)$.

Chương 6

6.1 (a) PDA đoán nhận $\{a^n b^m c^n \mid m, n \geq 1\}$ được định nghĩa như sau:

$A = (\{q_0, q_1\}, \{a, b\}, \{a, Z_0\}, \delta, q_0, Z_0, \varnothing\}$, trong đó δ được xác định như sau:

R_1: $\delta(q_0, a, Z_0) = \{(q_0, aZ_0)\}$

R_2: $\delta(q_0, a, a) = \{(q_0, aa)\}$

R_3: $\delta(q_0, b, a) = \{(q_1, a)\}$

R_4: $\delta(q_1, b, a) = \{(q_1, a)\}$

R_5: $\delta(q_1, a, a) = \{(q_1, \Lambda)\}$

R_6: $\delta(q_1, \Lambda, Z_0) = \{(q_1, \Lambda)\}$

Bắt đầu vào từ việc xếp các ký hiệu a vào kho đẩy xuống cho đến khi gặp b (luật R_1, R_2). Khi b xuất hiện, ôtômát thay đổi trạng thái nhưng vẫn không thay đổi kho đẩy xuống (luật R_3). Khi các ký hiệu b được đọc hết (luật R_4), sau đó xoá đi các ký hiệu a (luật R_5).

Vậy, $(q_0, a^n b^m a^n, Z_0) \vdash^* (q_1, \Lambda, Z_0) \vdash (q_1, \Lambda, \Lambda)$, nghĩa là $a^n b^m a^n \in N(A)$.

Sau đó chứng minh $N(A) = \{a^n b^m a^n \mid m, n \geq 1\}$.

(b) Ôtômát PDA để đoán nhận $\{a^n b^{2n} \mid n \geq 1\}$ được định nghĩa như sau:

$A = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, Z_0\}, \delta, q_0, Z_0, \varnothing\}$, trong đó δ được xác định bởi:

$\delta(q_0, a, Z_0) = \{(q_1, aZ_0)\}$, $\delta(q_1, a, a) = \{(q_1, aa)\}$, $\delta(q_1, b, a) = \{(q_2, a)\}$

$\delta(q_2, b, a) = \{(q_1, \Lambda)\}$, $\delta(q_1, \Lambda, Z_0) = \{(q_1, \Lambda)\}$.

(c) Ôtômát PDA để đoán nhận $\{a^m b^m c^n \mid m, n \geq 1\}$ được định nghĩa như sau:

$A = (\{q_0, q_1\}, \{a, b, c\}, \{Z_0, Z_1\}, \delta, q_0, Z_0, \varnothing\}$, trong đó δ được xác định bởi:

$\delta(q_0, a, Z_0) = \{(q_0, Z_1Z_0)\}$, $\delta(q_0, a, Z_1) = \{(q_0, Z_1Z_1)\}$,

$\delta(q_0, b, Z_1) = \{(q_1, \Lambda)\}$, $\delta(q_1, b, Z_1) = \{(q_1, \Lambda)\}$,

$\delta(q_1, c, Z_0) = \{(q_1, Z_0)\}$, $\delta(q_1, \Lambda, Z_0) = \{(q_1, \Lambda)\}$.

6.3 (a) $G = (\{S, S_1, S_2\}, \{a, b\}, P, S)$ để sinh ra $\{a^n b^n \mid n \geq 1\} \cup \{a^m b^{2m} \mid m \geq 1\}$, trong đó P có các qui tắc: $S \to S_1 \mid S_2$, $S_1 \to aS_1b \mid ab$, $S_2 \to aS_2bb \mid abb$.

Ôtômát PDA để đoán nhận L(G) được định nghĩa như sau:

$A = (\{q\}, \{a, b\}, \{S, S_1, S_2, a, b\}, \delta, q, S, \varnothing\}$, trong đó δ được xác định bởi:

$\delta(q, \Lambda, S) = \{(q, S_1), (q, S_2)\}$,

$\delta(q, \Lambda, S_1) = \{(q, aS_1b), (q, ab)\}$,

$\delta(q, \Lambda, S_2) = \{(q, aS_2bb), (q, abb)\}$,

$\delta(q, a, a) = \delta(q, b, b) = \{(q, \Lambda)\}$.

6.4 Ôtômát PDA để đoán nhận L được định nghĩa như sau:

$A = (\{q_0, q_1\}, \{a, b\}, \{Z_0, a, b\}, \delta, q_0, Z_0, \varnothing\}$, trong đó δ được xác định bởi:

$\delta(q_0, a, Z_0) = \{(q_0, aZ_0)\}$, $\delta(q_0, b, Z_0) = \{(q_0, bZ_0)\}$,

$\delta(q_0, a, b) = \{(q_0, ab)\}$, $\delta(q_0, b, a) = \{(q_0, ba)\}$,

$\delta(q_0, a, a) = \{(q_0, aa), (q_1, \Lambda)\}$,

$\delta(q_0, b, b) = \{(q_0, bb), (q_1, \Lambda)\}$,

$\delta(q_0, \Lambda, Z_0) = \{(q, aS_2bb), (q, abb)\}$,

$\delta(q_1, a, a) = \{(q_1, \Lambda)\}$, $\delta(q_1, b, b) = \{(q_1, \Lambda)\}$

$\delta(q_1, \Lambda, Z_0) = \{(q_1, \Lambda)\}$.

6.7 $M = (Q, \Sigma, \delta, q_0, F)$ là ôtômát hữu hạn đoán nhận ngôn ngữ tập chính qui. Định nghĩa PDA $A = (Q, \Sigma, \{Z_0\}, \delta_1, q_0, Z_0, F\}$, trong đó δ_1 được xác định bởi:

$\delta_1(q, a, Z_0) = \{(q', Z_0)\}$ nếu $\delta(q, a) = q'$.

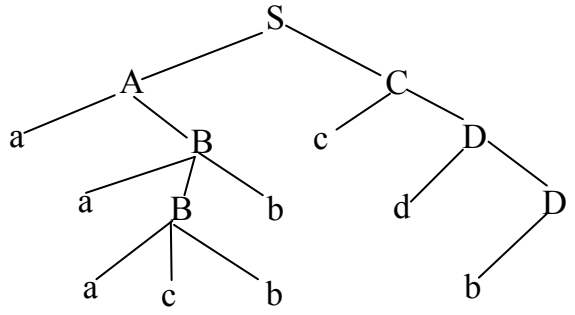
Dễ dàng chỉ ra rằng $T(M) = T(A)$.

6.8* (i) Văn phạm phi ngữ cảnh G để sinh ra $L = \{a^n cb^m cd^k \mid n > m > 0,\ k > 0\}$ có các qui tắc: $S \to AC$, $A \to aB$, $B \to aBb \mid aB \mid acb$, $C \to cD$, $D \to dD \mid d$.

(ii) Dẫn xuất để sinh ra $a^3cb^2cd^2$: $S \Rightarrow AC \Rightarrow aBC \Rightarrow a^2BbC \Rightarrow a^3cb^2C$

$\Rightarrow a^3cb^2cD \Rightarrow a^3cb^2cdD \Rightarrow a^3cb^2cd^2$

(iii) Cây cú pháp của từ $a^3cb^2cd^2$:



(iv) Ôtômát đẩy xuống M để đoán nhận L được định nghĩa như sau:

$M = (\{q\}, \{a, b, c, d\}, \{A, B, C, D, a, b, c, d\}, \delta, q, S, \varnothing\}$, với

$\delta(q, \Lambda, S) = \{(q, AC)\}$, $\delta(q, \Lambda, A) = \{(q, aB)\}$,

$\delta(q, \Lambda, B) = \{(q, ab), (q, acb), (q, aB)\}$, $\delta(q, \Lambda, C) = \{(q, cD)\}$,

$\delta(q, \Lambda, D) = \{(q, dD), (q, d)\}$,

$\delta(q, a, a) = \delta(q, b, b) = \delta(q, c, c) = \delta(q, d, d) = \{(q, \Lambda)\}$

Dễ dàng chỉ ra rằng N(M) = L(G).

(iv) $(q, a^3cb^2cd^2, S) \vdash (q, a^3cb^2cd^2, AC) \vdash (q, a^3cb^2cd^2, aBC)$

$\vdash (q, a^2cb^2cd^2, BC) \vdash (q, a^2cb^2cd^2, aBC) \vdash^* (q, \Lambda, \Lambda)$.

6.9* (i) Ngôn ngữ sinh $L(G) = \{wdw^T \mid w \in \{a, b, c\}^*, w^T$ là đảo ngược của w}

(ii) Ôtômát đẩy xuống M để đoán nhận L được định nghĩa như sau:

$M = (\{q\}, \{a, b, c, d\}, \{S, a, b, c, d\}, \delta, q, S, \varnothing\}$, với

$\delta(q, \Lambda, S) = \{(q, aSa), (q, bSb), (q, cSc), (q, d)\}$,

$\delta(q, a, a) = \delta(q, b, b) = \delta(q, c, c) = \delta(q, d, d) = \{(q, \Lambda)\}$

Dễ dàng chỉ ra rằng N(M) = L(G).

(iii) w_1 = aababbccdccbbabaa $\in L(G)$ vì

$S \Rightarrow aSa \Rightarrow aaSaa \Rightarrow aabSbaa \Rightarrow aabaSabaa \Rightarrow aababSbabaa$

$\Rightarrow aababbSbbabaa \Rightarrow aababbcScbbabaa \Rightarrow aababbccSccbbabaa$

$\Rightarrow aababbccdccbbabaa$.

Tương tự $w_1 \in L(M)$ vì

$(q, aababbccdccbbabaa, S) \vdash (q, aababbccdccbbabaa, aSa)$

$\vdash (q, ababbccdccbbabaa, Sa) \vdash (q, ababbccdccbbabaa, aSaa)$

$\vdash (q, babbccdccbbabaa, Saa) \vdash (q, babbccdccbbabaa, bSbaa)$

$\vdash (q, abbccdccbbabaa, Saa) \vdash (q, abbccdccbbabaa, aSabaa)$

$\vdash^* (q, \Lambda, \Lambda)$.

Nhưng $w_1 \notin L(G)$ vì không có dãy chuyển trạng thái nào để sinh ra w_1. Tương tự, không có dãy chuyển trạng thái nào trong M để chuyển w_1 về được trạng thái kết thúc, $w_1 \notin L(M)$.

Chương 7

7.1 $L(G) = \{a^{n+1}b^n \mid n \geq 0\}$. Với từ dạng $a^{n+1}b^n$, $A \to a$ được áp dụng ở bước cuối khi a đứng trước ab. Vậy $A \to a$ là qui tắc điều khiển khi và chỉ khi ký hiệu bên phải của a là b. Tương tự $A \to aAb$ là qui tắc điều khiển khi và chỉ khi ký hiệu bên phải của aAb là b và $S \to aAb$ là qui tắc điều khiển khi và chỉ khi ký hiệu bên phải của aAb là Λ. Từ đó suy ra G là LR(1) và không phải là LR(0).

7.2 Văn phạm G cho trước không phải là LR(k), $k \geq 0$. Giả sử G là LR(k), với k nào đó. Hãy xét dẫn xuất phải nhất của $01^{2k+1}2$ và 01^{2k+3}:

$$S \overset{*}{\underset{R}{\Rightarrow}} 01^kA1^k2 \underset{R}{\Rightarrow} 01^{2k+1}2 \Rightarrow \alpha\beta w \qquad (A7.1)$$

khi $\alpha = 01^k$, $\beta = A$ và $w = 1^k2$.

$$S \overset{*}{\underset{R}{\Rightarrow}} 01^{k+1}A1^{k+1}2 \underset{R}{\Rightarrow} 01^{2k+3}2 \Rightarrow \alpha'\beta'w' \qquad (A7.2)$$

với $\alpha' = 01^{k+1}$, $\beta' = A$, $w' = 1^{k+1}2$. Vì các xâu được tạo ra bởi 2k + 1 ký tự đầu tiên (lưu ý $|\alpha\beta| + k = 2k + 1$) của $\alpha\beta w$ và $\alpha'\beta'w'$ là một, $\alpha = \alpha'$, nghĩa là $01^k = 01^{k+1}$, dẫn đến mâu thuẫn.

7.3 Văn phạm G là nhập nhằng vì ab có hai cây dẫn xuất, do đó không có một số nguyên *k* nào đó để văn phạm này là LR(k).

7.4 Vì $a^nb^nc^n$ xuất hiện trong cả hai tập, do đó sẽ có hai cây dẫn xuất khác nhau. Vậy G là nhập nhằng, suy ra không phải là LR(k) với mọi *k* nguyên dương.

Chương 8

8.2 Dãy tính toán đối với xâu con 12 của 1213 là

$$q_1 1213 \vdash Bq_2 213 \vdash BBq_3 13.$$

Vì $\delta(q_3, 1)$ không xác định nên MT dừng. Đối với 2133, 312 thì máy Turing cho trước ở ví dụ 8.7 không bắt đầu được.

8.3 Xây dựng M có ba trang thái: q_0, q_1, q_f, trong đó q_0 là trạng thái bắt đầu được sử dụng để ghi nhận chẵn số lần số 1 đã gặp và q_1 được sử dụng để ghi nhận lẻ lần số 1 đã gặp, q_f là trạng thái kết thúc. M được xác định bởi bảng chuyển trạng thái như sau:

Trạng thái	0	1	B
$\rightarrow q_0$	$0Rq_0$	$1Rq_1$	BRq_f
q_1	$0Rq_1$	$1Rq_0$	
q_f			

8.5 Máy Turing để đoán nhận tất cả các Palindrrome có độ dài chẵn trên tập {0, 1} được xây dựng như sau:

(a) MT đọc ký tự vào từ băng vào (0 hoặc 1), xoá ký tự đó đi và chuyển sang trạng thái khác (q_1 hoặc q_2).

(b) MT đọc phần còn lại và không thay đổi các ký hiệu trên băng cho đến khi gặp dấu trắng B.

(c) Đầu đọc R/W chuyển sang trái. Nếu ký hiệu bên phải nhất phù hợp với ký hiệu bên trái nhất thì ký hiệu phải nhất được xoá đi, ngược lại MT dừng.

(d) R/W chuyển sang trái cho đến khi gặp B.

Bảng chuyển trạng thái mô tả hoạt động của MT:

Trạng thái hiện thời	Ký hiệu vào		
	0	1	B
→ q_0	BRq_1	BRq_2	BRq_1
q_1	$0Rq_1$	$1Rq_1$	BRq_3
q_2	$0Rq_1$	$1Rq_2$	BRq_4
q_3	BLq_5		
q_4		BLq_6	
q_5	$0Lq_5$	$1Lq_5$	BRq_0
q_6	$0Lq_6$	$1Lq_6$	BRq_0
q_7			

8.6 Giả sử bài toán giải được. Khi đó sẽ tồn tại thuật toán để quyết định được một từ cho trước có thuộc L hay không, nghĩa là có MT để đoán nhận L. Do vậy, sẽ tồn tại G để L(G) = L. Điều này suy ra $w \in L(G)$ khi và chỉ khi MT là dừng với w, nghĩa là bài toán dừng của MT là giải được, dẫn đến mâu thuẫn.

8.8 Giả sử $S = \{x \mid x \text{ là tập hợp và } x \notin x\}$. Nếu S là tập hợp thì $S \in S$ hoặc $S \notin S$. Nếu $S \notin S$, theo định nghĩa của S thì $S \in S$. Ngược lại, nếu $S \in S$ thì theo định nghĩa của S, $S \notin S$. Do đó ta không thể khẳng định $S \in S$ hay $S \notin S$.

8.9 Đặt $\Sigma = \{a\}$, $x = (x_1, \ldots, x_n)$ và $y = (y_1, \ldots, y_n)$, trong đó $x_i = a^{k_i}$, i = 1, 2, …, n và $y_j = a^{l_j}$, j = 1, 2, …, n. Khi đó

$(x_1)^{l_1} (x_2)^{l_2} \ldots (x_n)^{l_n} = (y_1)^{k_1} (y_2)^{k_2} \ldots (y_n)^{k_n}$. Bởi cả hai vế đều bằng $a^{\sum k_i l_i}$, nghĩa là PCP là giải được khi $|\Sigma| = 1$.

8.11 $x_1 = 01$, $y_1 = 011$, $x_2 = 1$, $y_2 = 10$, $x_3 = 1$ và $y_3 = 11$. Vì $|x_i| < |y_i|$, với i = 1, 2, 3.

Nên $x_{i_1} \ldots x_{i_m} \neq y_{i_1} \ldots y_{i_m}$, với mọi i_j. Suy ra PCP không có lời giải.

8.12 $x_1 = 0$, $y_1 = 01$, $x_2 = 110$, $y_2 = 000$, $x_3 = 001$, $y_3 = 10$. Ở đây tất cả các cặp (x_1,y_1), (x_2,y_2), (x_3,y_3) đều không có phần đầu (dãy con tiếp đầu ngữ) chung nhau. Vậy suy ra $x_{i_1} \ldots x_{i_m} \neq y_{i_1} \ldots y_{i_m}$, với mọi i_j. Suy ra rằng PCP với $S = \{(0, 10), (110, 0^3), (0^21, 10)\}$ không có lời giải.

Danh sách các từ viết tắt và thuật ngữ

A. Các ký hiệu

Ký hiệu	Ý nghĩa
$a \in A$	a thuộc tập A
$a \notin A$	a không thuộc tập A
$A \subseteq B$	A là tập con của B
$\varnothing$	Tập rỗng, tập trống
Λ	Phần tử rống, phần tử trống
$A \cup B$	Hợp của tập A và B
$A \cap B$	Giao của tập A và B
$A - B$	Hiệu của tập A và B
A^C	Phần bù của tập A
2^A	Tập tất cả các tập con của A
$A \times B$	Tích Đề Các của tập A và B
C_a	Lớp tương đương chứa a
R^+	Bao đóng bắc cầu của A
R^*	Bao đóng phản xạ, bắc cầu của A
$R_1 \circ R_2$	Hợp thành (ghép) của hai quan hệ R_1 và R_2
$\wedge$	Hội mệnh đề, phép "và" logic, "AND"
$\vee$	Tuyển mệnh đề, phép "hoặc logic, "OR"
$P \rightarrow Q$	Phép keo theo, P suy ra Q
$\neg$	Phép phủ định logic
$\equiv$	Phép đồng nhất
$\forall$	Lượng tử (lượng từ) phổ dụng: với mọi
$\exists$	Lượng tử (lượng từ) tồn tại
$\cong$	Quan hệ tương đương
$f: X \rightarrow Y$	Hàm f từ X sang Y
$\|x\|$	Độ dài (số phần tử, cỡ) của xâu (dãy) x,
$(Q, \Sigma, \delta, q_0, F)$	Một ôtômát hữu hạn
$\vdash$	Một phép chuyển trạng thái
(V_N, Σ, P, S)	Một văn phạm
$A \rightarrow \alpha$	Qui tắc, suy diễn trong văn phạm
$\underset{G}{\Rightarrow}$	Một dẫn xuất trong văn phạm G
$\overset{*}{\Rightarrow}$	Một dãy dẫn xuất của văn phạm
$\mathbf{R_1 + R_2}$	Hợp của các biểu thức chính qui $\mathbf{R_1}$ và $\mathbf{R_2}$
$\mathbf{R_1R_2}$	Ghép của các biểu thức chính qui $\mathbf{R_1}$ và $\mathbf{R_2}$
$\mathbf{R_1^*}$	Bao đóng (lặp) của các biểu thức chính qui $\mathbf{R_1}$ và $\mathbf{R_2}$

a	Biểu thức chính qui được biểu diễn bởi {a}
$(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$	Một ôtômát đẩy xuống
ID	Một mô tả, hiện trạng của PDA
P	Lớp các ngôn ngữ đoán nhận được bởi máy Turing đơn định với thời gian là đa thức,
NP	Lớp các ngôn ngữ đoán nhận được bởi máy Turing không đơn định với thời gian là đa thức,
$\preccurlyeq_p$	Phép dẫn đa thức

B. Danh sách các thuật ngữ viết tắt

Từ viết tắt	*Ý nghĩa*
MĐ	Tập các mệnh đề
T	Giá trị đúng (True)
F	Giá trị sai (False)
ÔTKĐĐ	Ôtômát hữu hạn không đơn định
ÔTĐĐ	Ôtômát hữu hạn đơn định
V_N	Tập các ký hiệu không kết thúc, các biến
Σ	Tập các chữ cái, các ký hiệu kết thúc
L(G)	Tập các xâu được sinh bởi văn phạm G
$£_0$	Lớp tất cả các ngôn ngữ loại 0
$£_1$	Lớp tất cả các ngôn ngữ loại 1
$£_2$	Lớp tất cả các ngôn ngữ phi ngữ cảnh (loại 2)
$£_3$	Lớp tất cả các ngôn ngữ chính qui (loại 3)
CNF	Dạng chuẩn Chomsky
GNF	Dạng chuẩn Greibach
PDA	Ôtômát đẩy xuống
PDS	Kho đẩy xuống
N(A)	Tập đoán nhận bởi PDA với kho PDS rỗng
T(M)	Tập các xâu đoán nhận được bởi ôtômát M
LL(k)	Tập con của văn phạm phi ngữ cảnh, đọc trước k ký tự ở phía trước
LR(k)	Tập con của văn phạm phi ngữ cảnh, đọc các ký tự từ trái qua phải
LBA	Ôtômát tuyến tính giới nội

PCP	Bài toán tương ứng Post
CSG	Văn phạm cảm ngữ cảnh, văn phạm loại 1
CSL	Ngôn ngữ cảm ngữ cảnh, ngôn ngữ loại 1
SAT	Bài toán thoả của công thức mệnh đề

C. Danh sách các thuật ngữ

Tiếng Anh	*Tiếng Việt*
Absorption law	Luật hấp thụ
Associative laws	Luật kết hợp
Binary operation	Phép toán nhị nguyên
Bottom-up parsing	Phân tích cú pháp dưới lên trên
Commutative law	Luật giao hoán
Commutative ring	Vành giao hoán
Commutative ring with identity	Vành giao hoán có đơn vị
Computable function	Hàm tính được
Concatenation	Ghép xâu
Conjunctive normal form	Dạng chuẩn hội
Constructive dilemma	Định đề kiến thiết
Context-free grammar	Văn phạm phi ngữ cảnh, văn phạm loại 2
Context-sensitive grammar	Văn phạm cảm ngữ cảnh, văn phạm loại 1
Derivation	Dẫn xuất, qui tắc
Derivation tree (Parse tree)	Cây dẫn xuất, cây phân tích cú pháp
Deterministic finite automaton	Ôtômát hữu hạn đơn định, ÔTĐĐ
Disjunctive normal form	Dạng chuẩn tuyển
Disjunctive syllogism	Tam đoạn luận
Destructive dilemma	Định đề đối thiết
Distributive law	Luật phân phối
Emigroup	Nửa nhóm
Field	Trường
Finite automaton	Ôtômát hữu hạn (trạng thái)
Formula SATisfiability	Bài toán thoả của công thức mệnh đề
Group	Nhóm
Handle	Cán điều khiển, điều khiển
Hypothetical syllogism	Suy luận bắc cầu
Idempotent law	Luật luỹ đẳng

Instance Description	Thể hiện, mô tả hiện trạng, ID
Left to Right	Trái qua phải, LR
Linear Bounded Automata	Ôtômát tuyến tính giới nội
Monotonic	Đơn điệu
Nondeterministic finite automaton	Ôtômát hữu hạn không đơn định, ÔTKĐĐ
Normal Form	Dạng chuẩn
NP-completeness	NP-đầy đủ
Paradox	Nghịch lý
Post Correspondence Problem	Bài toán tương ứng Post, PCP
Predicate	Tân từ, vị từ
Principal conjunctive normal form	Dạng chuẩn hội chính tắc
Principal disjunctive normal form	Dạng chuẩn tuyển chính tắc
Proposition (Statement)	Mệnh đề
Pumping lemma	Bổ đề Bơm
Pushdown automaton	Ôtômát đẩy xuống
Pushdown store	Kho đẩy xuống, ngăn xếp
Quantifier	Lượng tử
Random Access Machine	Máy truy cập ngẫu nhiên, RAM
Recursive	Đệ qui
Recursively enumerable	Đệ qui đếm được, đệ qui kể được
Reflexive	Phản xạ
Regular Expression	Biểu thức chính qui
Regular grammar	Văn phạm chính qui, văn phạm loại 3
Regular Set	Tập chính qui
Ring	Vành
Rule of inference	Qui tắc suy diễn
Self-embedding	Tự nhúng
Shift register	Thanh ghi dịch chuyển
String	Xâu, dãy ký tự
Symmetric	Đối xứng
Tautology	Công thức hằng đúng
Top-down parsing	Phân tích cú pháp trên xuống
Transition function	Hàm chuyển trạng thái, hàm chuyển
Transitive	Bắc cầu
Transition system	Hệ thống chuyển trạng thái, hay chuyển đổi
Well-formed formula	Công thức (mệnh đề), biểu thức logic

Tài liệu tham khảo

[1] Ding-Zhu Du, Ker-I Ko, *Theory of Computational Complexity,* John Wiley & Sons, INC, 2000.

[2] Hopcroft J. and Ullman J., *Formal languages and their relation to automata*, Addison Wesley, London, 1969.

[3] Johnsonbaugh R., *Discrete Mathematics,* Macmillan Publishing Company, 1992.

[4] Mishra K. and Chandrasekaran, *Theory of Computer Science (Automata, Languages and Computation),* Prentice Hall, 2001.

[5] Nguyễn Văn Ba, *Ngôn ngữ hình thức,* ĐH Bách Khoa Hà Nội, 1997.

[6] Rosen K. H., *Discrete Mathematics and Its Application,* Mc Graw- Hill, NY 1994.