

Chương VII

MÁY TURING

Nội dung chính : Trong chương này, ta sẽ xét thêm một loại máy trừu tượng khác - máy Turing (TM - Turing Machines). Chúng có khả năng đoán nhận được lớp ngôn ngữ lớn hơn lớp ngôn ngữ phi ngữ cảnh. Đây còn là một mô hình của sự tính toán, mô hình của các thủ tục hiệu quả, là nền tảng cho quá trình xử lý của máy tính hiện đại, được giới thiệu bởi Alan Turing vào năm 1936. Nhờ đó, các khái niệm về "sự tính được", "sự giải được" được xác định một cách rõ ràng trên cơ sở sự xuất hiện của một số hàm không tính được, các bài toán không giải được.

Mục tiêu cần đạt: Cuối chương, sinh viên cần phải nắm vững:

- Khái niệm TM, định nghĩa và các thành phần.
- Các kỹ thuật thiết kế TM.
- Một số biến dạng TM từ mô hình chuẩn.
- Xây dựng TM dùng nhận dạng ngôn ngữ hoặc tính toán các hàm số nguyên đơn giản được biểu diễn trong hệ nhất phân.
- Các tính chất của lớp ngôn ngữ được chấp nhận bởi TM.

Kiến thức cơ bản: Để tiếp thu tốt nội dung của chương này, sinh viên cần hiểu rõ cách thiết kế các hàm chuyển trạng thái trên mô hình máy tính toán; ý tưởng thiết kế một số thuật toán đơn giản trên tập hợp số, ...

Tài liệu tham khảo :

[1] John E. Hopcroft, Jeffrey D.Ullman – *Introduction to Automata Theory, Languages and Computation* – Addison – Wesley Publishing Company, Inc – 1979 (**Chapter 7 : Turing Machines**)

[2] Peter Linz – *An Introduction to Formal Languages and Automata* – D.C. Heath and Company – 1990.

[3] **David Barker-Plummer** - Stanford Encyclopedia of Philosophy - *Turing Machines*:

<http://plato.stanford.edu/entries/turing-machine/>

[4] **Turing Machines implemented in JavaScript:**

<http://www.turing.org.uk/turing/scrapbook/tmjjava.html>

[5] By Jon Barwise and John Etchemendy -*Turing Machines*:

<http://www-csli.stanford.edu/hp/Turing1.html>

I. MÔ HÌNH MÁY TURING (TM)

Một mô hình hình thức cho một thủ tục hiệu quả sẽ có những đặc tính cụ thể. Đầu tiên, mỗi thủ tục sẽ được mô tả một cách hữu hạn. Tiếp đó, thủ tục sẽ được phân thành một số bước độc lập, mà mỗi bước thực thi một vấn đề. Nguyên tắc này cũng được hình thức trong mô hình máy Turing.

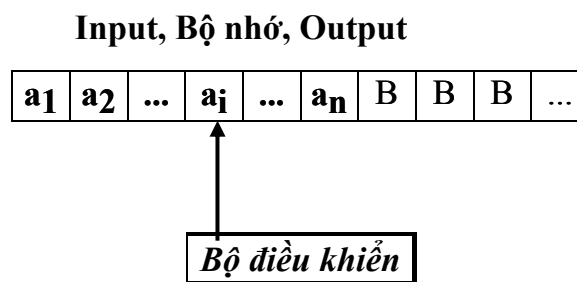
Máy Turing có một băng nhớ, dùng để ghi mọi loại dữ liệu (dữ liệu nhập, dữ liệu dùng cho việc điều khiển tương tự như một chương trình máy tính và các kết quả trung gian khi làm việc). Với một bộ điều khiển chứa một số hữu hạn trạng thái, TM cũng như các ô tô khác, làm việc theo lối "ngắt quãng" theo từng bước chuyển.

1.1. Mô tả TM

Máy Turing có rất nhiều dạng đồng khả năng, nghĩa là có nhiều mô hình và định nghĩa khác nhau cho máy Turing nhưng tất cả chúng đều tương đương nhau. Song, nói chung mô hình cơ bản của một máy Turing gồm :

- Một bộ điều khiển hữu hạn.
- Một băng được chia thành các ô.
- Một **đầu đọc-viết**, mỗi lần đọc có thể duyệt qua một ô trên băng để đọc hay viết ký hiệu.

Mỗi ô có thể giữ được một ký hiệu trong số hữu hạn các ký hiệu băng (các ký hiệu được phép viết trên băng). Khởi đầu xem như n ô bên trái của băng ($n \geq 0$) giữ chuỗi nhập (input), chuỗi nhập là một chuỗi các ký tự được chọn từ một tập hợp con của tập hợp các ký hiệu băng, tập hợp con này gọi là tập các ký hiệu nhập. Phần còn lại của băng coi như có vô hạn khoảng trống, ký hiệu B (Blank), B là một ký hiệu đặc biệt của băng nhưng không phải là ký hiệu nhập.



Hình 7.1 - Mô tả một TM

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

- 1) Chuyển trạng thái
- 2) In một ký hiệu trên băng tại ô đang duyệt (nghĩa là thay ký hiệu đọc được trên băng bằng ký hiệu nào đó)
- 3) Dịch chuyển đầu đọc-viết (sang trái (L), sang phải (R) hoặc đứng yên(\emptyset))

Câu hỏi :



So sánh cơ chế máy Turing với hai dạng ô tô máy đã khảo sát trong các chương trước (ô tô máy hữu hạn FA và ô tô máy đẩy xuống PDA) ? Nêu những điểm khác biệt quan trọng trong nguyên tắc nhận dạng ngôn ngữ ?

1.2. Định nghĩa

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

Định nghĩa: TM là một hệ thống $M(Q, \Sigma, \Gamma, \delta, q_0, B, F)$, trong đó:

- . Q : tập hữu hạn các trạng thái.
- . Σ : bộ ký hiệu nhập.
- . Γ : tập hữu hạn các ký tự được phép viết trên băng.
- . B : ký hiệu thuộc Γ dùng chỉ khoảng trống trên băng (Blank).
- . δ : hàm chuyển ánh xạ : $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, \emptyset\}$
(δ có thể không xác định với một vài đối số)
- . $q_0 \in Q$ là trạng thái bắt đầu
- . $F \subseteq Q$ là tập các trạng thái kết thúc

Hình thái TM (Instantaneous description - ID)

Một hình thái của máy Turing M được cho bởi $\alpha_1 q \alpha_2$, trong đó q là trạng thái hiện hành của M ; $\alpha_1 \alpha_2 \in \Gamma^*$ là nội dung của băng tính từ đầu băng cho tới ký hiệu khác Blank bên phải nhất của băng. Giả sử Q và Γ rời nhau: đầu đọc đang đọc ký hiệu bên trái nhất của α_2 hoặc nếu $\alpha_2 = \epsilon$ thì đầu đọc đang đọc Blank.

Hàm chuyển

Ta định nghĩa một phép chuyển trạng thái của TM như sau :

Đặt $X_1 X_2 \dots X_{i-1} q X_i \dots X_n$ là một ID.

+ Giả sử $\delta(q, X_i) = (p, Y, L)$, trong đó:

- 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.

- Nếu $i > 1$ ta viết :

$$X_1 X_2 \dots X_{i-1} \mathbf{q} X_i \dots X_n \vdash_M X_1 X_2 \dots X_{i-2} \mathbf{p} X_{i-1} \mathbf{Y} X_{i+1} \dots X_n$$

+ Tương tự $\delta(\mathbf{q}, X_i) = (\mathbf{p}, \mathbf{Y}, \mathbf{R})$ thì ta viết :

$$X_1 X_2 \dots X_{i-1} \mathbf{q} X_i \dots X_n \vdash_M X_1 X_2 \dots X_{i-1} \mathbf{Yp} X_{i+1} \dots X_n$$

+ Tương tự $\delta(\mathbf{q}, X_i) = (\mathbf{p}, \mathbf{Y}, \emptyset)$ thì ta viết :

$$X_1 X_2 \dots X_{i-1} \mathbf{q} X_i \dots X_n \vdash_M X_1 X_2 \dots X_{i-1} \mathbf{pY} X_{i+1} \dots X_n$$

Chú ý rằng nếu $i - 1 = n$ thì chuỗi $X_i \dots X_n$ là rỗng và vế phải dài hơn vế trái, nghĩa là TM M mở rộng chuỗi ký hiệu trên băng.

Nếu hai ID được quan hệ nhau bởi \vdash_M thì ta nói ID thứ hai là kết quả của ID thứ nhất bằng một lần chuyển, một bước áp dụng hàm chuyển (hoặc nói cái thứ hai thu được từ cái thứ nhất bằng một lần chuyển). Nếu một ID thu được từ ID khác bằng một số lần chuyển (có thể bằng 0) thì ta ký hiệu quan hệ là \vdash_M^* . Ta cũng có thể bỏ đi ký hiệu M trong cách viết các quan hệ trên nếu không có nhầm lẫn.

Ngôn ngữ được chấp nhận bởi TM

Ký hiệu $L(M)$: tập hợp các chuỗi trong Γ^* là nguyên nhân đưa TM M đi vào trạng thái kết thúc khi đã thực hiện việc thay thế từ bên trái các ký hiệu trên băng của M với trạng thái bắt đầu q_0 . Một cách hình thức, ta định nghĩa tập hợp ngôn ngữ được chấp nhận bởi TM $M(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ là tập

$$L(M) = \{ w \mid w \in \Gamma^* \text{ và } q_0 w \vdash_M^* \alpha_1 p \alpha_2 \text{ với } p \in F \text{ còn } \alpha_1 \alpha_2 \in \Gamma^* \}$$

Cho TM nhận diện một ngôn ngữ L là cho lần lượt các từ của L vào TM xem TM có chấp nhận từ đó không. TM sẽ dừng và đi vào một trong những trạng thái kết thúc $\in F$ (không có phép chuyển kế tiếp) khi từ được chấp nhận, nhưng nếu TM không chấp nhận một từ nào đó thì TM có thể ngừng ở một trạng thái $\notin F$ hoặc cũng có thể nó chạy mãi mà không dừng lại.

Thí dụ 7.1 : Thiết kế TM chấp nhận ngôn ngữ $L = \{ 0^n 1^n \mid n \geq 1 \}$

Khởi đầu TM chứa $0^n 1^n$ bên trái nhất trên băng sau đó là vô hạn khoảng trống Blank. TM lặp lại quá trình sau:

- M thay 0 bên trái nhất bằng X rồi chuyển sang phải tới 1 trái nhất, TM thay 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 TM gặp Blank thì TM dừng và không chấp nhận input. Tương tự, khi TM đã thay hết 0 bằng X và kiểm tra còn 1 trên băng thì TM cũng dừng và không chấp nhận input.

- TM chấp nhận input nếu như cũng không còn ký hiệu 1 nào nữa trên băng.

Đặt TM M $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ với các thành phần :

$Q = \{q_0, q_1, q_2, q_3, q_4\}$; $\Sigma = \{0, 1\}$; $\Gamma = \{0, 1, X, Y, B\}$ 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 M tìm thấy 1 nó thay 1 bằng Y rồi đi vào trạng thái q_2 . Trạng thái q_2 đưa M 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 M tiến sang phải trong trạng thái q_1 , nếu B hoặc X được tìm thấy trước 1 thì input bị loại bỏ (không chấp nhận) vì có chứa nhiều ký hiệu 0 hơn 1 hoặc input 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 thành X mà chỉ gặp Y thì TM 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 TM sẽ đi vào q_4 (trạng thái kết thúc) để chấp nhận input. Ngược lại input bị loại bỏ.

Hàm chuyển δ được cho trong bảng sau :

δ Trạng thái	Ký hiệu				
	0	1	X	Y	B
q_0	(q_1, X, R)	-	-	(q_3, Y, R)	-
q_1	$(q_1, 0, R)$	(q_2, Y, L)	-	(q_1, Y, R)	-
q_2	$(q_2, 0, L)$	-	(q_0, X, R)	(q_2, Y, L)	-
q_3	-	-	-	(q_3, Y, R)	(q_4, B, \emptyset)
q_4	-	-	-	-	-

Các phép chuyển hình thái của TM M trên input 0011 :

$q_00011 \vdash Xq_1011 \vdash X0q_111 \vdash Xq_20Y1 \vdash q_2X0Y1 \vdash Xq_00Y1 \vdash XXq_1Y1 \vdash XXYq_11 \vdash XXq_2YY \vdash Xq_2XYY \vdash XXq_0YY \vdash XXYq_3Y \vdash XXYq_3Y \vdash XXYq_4$

Nhận xét: Như vậy, ta có thể dễ dàng thấy, TM khác với một ô-tô-mát hữu hạn ở chỗ đầu đọc-viết có thể dịch chuyển tự do trên băng, không những đọc mà còn có khả năng viết trên băng và vùng làm việc còn có thể mở rộng theo yêu cầu phát sinh. TM khác với ô-tô-mát đẩy xuống ở chỗ nó không dùng thêm Stack như một bộ giữ nhớ mà viết các ký hiệu cần ghi nhớ ngay trên băng.

II. NGÔN NGỮ VÀ "HÀM TÍNH ĐƯỢC"

Ngôn ngữ được chấp nhận bởi một máy Turing được gọi là ngôn ngữ **đệ qui liệt kê - recursively enumerable (r.e)**. Đó là một lớp ngôn ngữ rất rộng, nó thực sự chứa ngôn ngữ phi ngữ cảnh CFL và một số ngôn ngữ mà không thể xác định các thành phần một cách máy móc. Nếu $L(M)$ là một ngôn ngữ như vậy thì bất kỳ một máy Turing nào nhận diện $L(M)$ cũng sẽ không dừng trên một số input không thuộc $L(M)$. Nhưng

nếu một chuỗi $w \in L(M)$ thì chắc chắn TM dừng, tuy nhiên TM sẽ chạy bao lâu trên input thì chúng ta không thể biết được và ta cũng không biết chắc chắn liệu TM có dừng lại hay không. Một cách thuận lợi và có ý nghĩa hơn là xét một lớp con của lớp ngôn ngữ đệ qui liệt kê, trong đó mọi ngôn ngữ đều được chấp nhận bởi ít nhất một máy Turing dừng trên mọi input. Lớp ngôn ngữ này gọi là lớp ngôn ngữ **đệ qui - recursive sets**.

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, \dots, i_k thì ta viết lần lượt các số nguyên này trên băng của TM ngăn cách nhau bởi 1, nghĩa là input có dạng $0^{i_1}10^{i_2}1 \dots 10^{i_k}$. Nếu TM dừng (chấp nhận hoặc không chấp nhận input) với băng 0^m thì ta nói $f(i_1, i_2, \dots, i_k) = m$.

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, \dots, 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 liệt kê 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 TM sẽ dừng trên mọi input.

Thí dụ 7.2 : Thiết kế máy Turing tính toán phép trừ riêng

Ta định nghĩa phép trừ riêng (proper subtraction) như sau :

$$f(m, n) = m \setminus n = \begin{cases} m - n & \text{nếu } m \geq n \\ 0 & \text{nếu } m < n \end{cases}$$

. Input : $0^m 10^n$

. Output : $0^{m \setminus n}$

M lặp lại việc thay thế lần lượt từng số 0 ở đầu băng bằng B rồi tiến sang phải, ra sau 1 tìm 0 và thay 0 này bằng 1. M 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. M dừng nếu :

i) Khi sang phải tìm 0 bên phải, M gặp B. Lúc này M đã thay n số 0 bên phải chuỗi input $0^m 10^n$ thành 1 và $n + 1$ số 0 bên trái thành B, trường hợp này xảy ra khi trong chuỗi input có $m > n$. Do vậy M 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, M 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 $n \leq m$. Khi đó, M 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 TM như sau: $M(\{q_0, q_1, \dots, q_6\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_6\})$

M sẽ bắt đầu bằng $0^m 10^n$ trên băng và kết thúc với $0^{m \setminus n}$ trên băng. Các phép chuyển trạng thái được định nghĩa như sau :

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

M thay 0 đầu băng bởi B.

$$2) \quad \delta(q_1, 0) = (q_1, 0, R)$$

$$\delta(q_1, 1) = (q_2, 1, R)$$

M di chuyển sang phải qua 0 tìm 1.

$$3) \quad \delta(q_2, 1) = (q_2, 1, R)$$

$$\delta(q_2, 0) = (q_3, 1, L)$$

M di chuyển sang phải vượt qua 1 đến khi gặp 0, đổi 0 thành 1.

$$4) \quad \delta(q_3, 0) = (q_3, 0, L)$$

$$\delta(q_3, 1) = (q_3, 1, L)$$

$$\delta(q_3, B) = (q_0, B, R)$$

M dịch trái tới khi gặp B, trở về trạng thái q_0 và bắt đầu một vòng lặp mới.

$$5) \quad \delta(q_2, B) = (q_4, B, L)$$

$$\delta(q_4, 1) = (q_4, B, L)$$

$$\delta(q_4, 0) = (q_4, 0, L)$$

$$\delta(q_4, B) = (q_6, 0, \emptyset)$$

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: TM 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 M đi vào trạng thái kết thúc q_6 và dừng.

$$6) \quad \delta(q_0, 1) = (q_5, B, R)$$

$$\delta(q_5, 0) = (q_5, B, R)$$

$$\delta(q_5, 1) = (q_5, B, R)$$

$$\delta(q_5, B) = (q_6, B, \emptyset)$$

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: TM 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 TM tính toán phép trừ $2 \setminus 1$ (tức input 0010) như sau :

$q_0 0010 \vdash B q_1 010 \vdash B 0 q_1 10 \vdash B 0 1 q_2 0 \vdash B 0 q_3 11 \vdash B q_3 0 11 \vdash q_3 B 0 11 \vdash B q_0 0 11 \vdash$
 $BB q_1 11 \vdash BB 1 q_2 1 \vdash BB 1 1 q_2 \vdash BB 1 q_4 1 \vdash BB q_4 1 \vdash B q_4 \vdash B q_6 0$

Nếu cho TM tính toán $1 \setminus 2$ (tức input 0100) :

$q_0 0100 \vdash B q_1 100 \vdash B 1 q_2 00 \vdash B q_3 110 \vdash q_3 B 110 \vdash B q_0 110 \vdash BB q_5 10 \vdash BBB q_5 0 \vdash$
 $BBBB q_5 \vdash BBBB q_6$

III. CÁC KỸ THUẬT XÂY DỰNG MÁY TURING

Việc xây dựng máy Turing bằng cách viết (liệt kê) tất cả các hàm chuyển của nó trên băng nhập có thể là một công việc đơn điệu. Để mô tả đầy đủ cách xây dựng máy Turing, ta cần một vài công cụ "cấp cao" hơn. Phần này sẽ trình bày một số công cụ tổng quát :

3.1. Lưu trữ trong bộ điều khiển (Storage in the finite control)

Bộ điều khiển có thể dùng để lưu trữ một lượng hữu hạn thông tin. Để làm như thế, ta viết mỗi trạng thái như là một cặp các phần tử: một thành phần để điều khiển, thành phần kia lưu giữ 1 ký hiệu. Chú ý rằng, đây chỉ là một cách mở rộng trên khái niệm chứ không thay đổi định nghĩa máy Turing.

Thí dụ 7.3 : Xét máy Turing M nhận vào ký hiệu đầu tiên trên chuỗi nhập (viết trên bộ chữ cái $\{0, 1\}$), lưu trữ vào bộ điều khiển và kiểm tra rằng ký hiệu này không có xuất hiện ở vị trí nào khác trên chuỗi nữa hay không ?.

Ta xây dựng TM M ($Q, \{0, 1\}, \{0, 1, B\}, \delta, [q_0, B], B, F$), trong đó tập trạng thái Q bao gồm các trạng thái dạng một cặp thành phần $\{q_0, q_1\} \times \{0, 1, B\}$, tức là Q gồm chứa các trạng thái $[q_0, 0], [q_0, 1], [q_0, B], [q_1, 0], [q_1, 1]$ và $[q_1, B]$. Trong mỗi cặp này thành phần thứ nhất ghi trạng thái điều khiển, thành phần thứ hai ghi nhớ ký hiệu. Ta định nghĩa hàm chuyển δ như sau:

$$1) \quad \delta([q_0, B], 0) = ([q_1, 0], 0, R)$$

$$\delta([q_0, B], 1) = ([q_1, 1], 1, R)$$

Bắt đầu từ trạng thái $[q_0, B]$, TM đọc và lưu trữ ký hiệu đầu tiên trên băng vào thành phần thứ hai trong bộ điều khiển.

$$2) \quad \delta([q_1, 0], 1) = ([q_1, 0], 1, R)$$

$$\delta([q_1, 1], 0) = ([q_1, 1], 0, R)$$

Nếu các ký hiệu được đọc tiếp theo không giống với ký hiệu đang lưu trữ thì tiếp tục di chuyển sang phải.

$$3) \quad \delta([q_1, 0], B) = ([q_1, B], 0, \emptyset)$$

$$\delta([q_1, 1], B) = ([q_1, B], 0, \emptyset)$$

M đi vào trạng thái kết thúc $[q_1, B]$ khi gặp Blank.

M sẽ đi vào trạng thái kết thúc nếu nó tiến đến gặp ký hiệu B mà không có ký hiệu nào giống với ký hiệu đầu tiên đang được lưu trữ trong bộ điều khiển. Vậy nếu M tiến đến B ở trạng thái $[q_1, 0]$ hoặc $[q_1, 1]$ thì input được chấp nhận. Ngược lại, ở trạng thái $[q_1, 0]$ và gặp 0 hoặc ở trạng thái $[q_1, 1]$ và gặp 1 thì M dừng và không chấp nhận chuỗi input vì không có hàm chuyển trạng thái để xác định các bước chuyển này.

Một cách tổng quát, ta có thể xem bộ điều khiển gồm k thành phần trong đó một thành phần giữ trạng thái điều khiển và các thành phần kia (k-1 thành phần) dùng lưu giữ thông tin.

3.2. Nhiều rãnh trên băng (Multiple tracks)

Một cách mở rộng khác, ta cũng có thể xem băng của TM được chia thành k thành phần, với $k > 1$ và hữu hạn. Một ký hiệu trên băng được xét là một bộ gồm k ký hiệu, mỗi ký hiệu nằm trên một rãnh.

Thí dụ 7.4 : Thiết kế TM nhận vào một số nguyên n (viết ở dạng nhị phân) và kiểm tra xem đó có phải là số nguyên tố hay không ?

Ta dùng băng 3 rãnh như hình 7.2 với nguyên tắc sau :

Số n ở dạng nhị phân được đưa vào trên rãnh 1 và được bao bởi cặp dấu φ và $\$$. Như vậy các ký hiệu được phép ghi trên băng là $[\varphi, B, B]$, $[0, B, B]$, $[1, B, B]$ và $[\$, B, B]$. Các ký hiệu này tương ứng với φ , 0, 1, $\$$ khi xem chúng là ký hiệu nhập. Ký hiệu Blank là $[B, B, B]$.

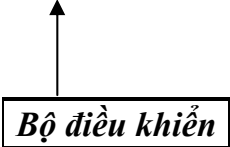
Viết số 2 dạng nhị phân trên rãnh 2 (tức 10)

Chép rãnh 1 vào rãnh 3 sau đó lấy rãnh 3 trừ rãnh 2 nhiều lần nhất có thể được (thực hiện việc chia số cần kiểm tra cho số trên rãnh 2, lấy phần dư)

Xét số còn lại (số dư) :

- Nếu số còn lại là 0 thì input không là số nguyên tố (vì nó chia hết cho số trên rãnh 2)
- Nếu số còn lại khác 0 thì tăng số trên rãnh 2 thêm một đơn vị: nếu số trên rãnh 2 bằng số trên rãnh 1 (số n) thì input n là số nguyên tố vì n đã không chia hết cho bất kỳ số nào từ 2 đến $n-1$. Nếu số trên rãnh 2 nhỏ hơn số trên rãnh 1 thì ta lặp lại quá trình trên với số mới trên rãnh 2.

φ	1	0	1	1	1	1	$\$$	B	B
B	B	B	B	1	0	1	B	B	B
B	1	0	0	1	0	1	B	B	B



Hình 7.2 - TM với băng 3 rãnh

Hình 7.2 trên mô tả một TM với $k = 3$, kiểm tra số $n = 47$ viết trên rãnh 1 dưới dạng nhị phân, TM đang thực hiện phép chia 47 cho 5. Nó đã trừ 2 lần số 5 vào số 47, vậy ở rãnh 3 hiện đang có số 37.

3.3. Đánh dấu ký hiệu (Checking off symbols)

Kỹ thuật đánh dấu thường dùng để nhận diện các ngôn ngữ được định nghĩa bằng cách lặp lại chuỗi chẳng hạn như $\{ww \mid w \in \Sigma^*\}$; $\{wcy \mid w, y \in \Sigma^*, w \neq y\}$ hoặc $\{ww^R \mid w \in \Sigma^*\}$ hoặc các ngôn ngữ có độ dài các chuỗi con cần được so sánh, như $\{a^i b^i \mid i \geq 1\}$ hoặc $\{a^i b^j c^k \mid i = j \text{ hoặc } j = k\}$.

Ta dùng một rãnh mở rộng trên băng để giữ ký hiệu đánh dấu \surd . Ký hiệu \surd xuất hiện khi ký hiệu trên rãnh ngay bên dưới nó đã hoặc đang được xét bởi TM.

Thí dụ 7.5 : Xét máy Turing $M (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ nhận diện ngôn ngữ L có dạng $\{wcw \mid w \in (a+b)^+\}$ với các thành phần như sau :

$Q = \{[q, d] \mid q = q_1, \dots, q_9 \text{ và } d = a, b \text{ hoặc } B\} = \{q_1, \dots, q_9\} \times \{a, b, B\}$ (thành phần thứ hai của các trạng thái dùng để lưu trữ ký hiệu nhập)

$\Sigma = \{[B, d] \mid d = a, b, c\}$ (ký hiệu nhập $[B, d]$ được xác định bởi d)

$\Gamma = \{[X, d] \mid X = B \text{ hoặc } \surd; d = a, b, c \text{ hoặc } B\}$.

$q_0 = [q_1, B]$

$B = [B, B]$ được định nghĩa là B , ký hiệu Blank.

$F = \{[q_9, B]\}$.

Với $d = a$ hoặc b ; $e = a$ hoặc b , ta định nghĩa hàm chuyển δ như sau:

1) $\delta([q_1, B], [B, d]) = ([q_2, d], [\surd, d], R)$

M đánh dấu ký hiệu được duyệt trên băng, lưu trữ vào bộ điều khiển và dịch chuyển sang phải.

2) $\delta([q_2, d], [B, e]) = ([q_2, d], [B, e], R)$

M tiếp tục dịch phải trên các ký hiệu chưa đánh dấu và tìm c .

3) $\delta([q_2, d], [B, c]) = ([q_3, d], [B, c], R)$

Khi tìm thấy c , M đi vào trạng thái mà thành phần đầu tiên là q_3 .

4) $\delta([q_3, d], [\surd, e]) = ([q_3, d], [\surd, e], R)$

M dịch phải qua các ký hiệu đã đánh dấu.

5) $\delta([q_3, d], [B, d]) = ([q_4, B], [\surd, d], L)$

M gặp ký hiệu chưa đánh dấu. Nếu ký hiệu chưa đánh dấu giống với ký hiệu đang lưu trong bộ điều khiển thì M đánh dấu rồi dịch trái. Nếu ký hiệu không giống ký hiệu lưu trong bộ điều khiển thì M không dịch chuyển nữa và không chấp nhận input. M cũng dừng nếu ở trạng thái q_3 và gặp ký hiệu $[B, B]$ trước khi gặp ký hiệu chưa đánh dấu.

6) $\delta([q_4, B], [\surd, d]) = ([q_4, B], [\surd, d], L)$

M dịch trái trên các ký hiệu đã đánh dấu.

7) $\delta([q_4, B], [B, c]) = ([q_5, B], [B, c], L)$

M gặp ký hiệu c .

8) $\delta([q_5, B], [B, d]) = ([q_6, B], [B, d], L)$

Nếu ký hiệu ngay bên trái c chưa được đánh dấu thì M tiến sang trái để tìm ký hiệu bên phải nhất đã được đánh dấu.

9) $\delta([q_6, B], [B, d]) = ([q_6, B], [B, d], L)$

M tiếp tục dịch chuyển sang trái.

10) $\delta([q_6, B], [\surd, d]) = ([q_1, B], [\surd, d], R)$

M gặp ký hiệu đã đánh dấu, nó dịch phải để lấy ký hiệu chưa đánh dấu bên cạnh và tiếp tục vòng lặp so sánh. Khi đó, thành phần thứ 1 lại trở thành q_1 .

11) $\delta([q_5, B], [\surd, d]) = ([q_7, B], [\surd, d], R)$

M ở trạng thái $[q_5, B]$ ngay sau khi vượt sang trái c . Nếu ký hiệu xuất hiện ngay trước c đã được đánh dấu thì tất cả các ký hiệu trước c đều đã được đánh dấu. M

phải kiểm tra xem bên phải c còn có ký hiệu nào chưa được đánh dấu hay không. Nếu không còn ký hiệu nào thì M chấp nhận input.

$$12) \quad \delta([q_7, B], [B, c]) = ([q_8, B], [B, c], R)$$

M dịch sang phải c.

$$13) \quad \delta([q_8, B], [\surd, d]) = ([q_8, B], [\surd, d], R)$$

M tiếp tục dịch sang phải trên các ký hiệu đã được đánh dấu.

$$14) \quad \delta([q_8, B], [B, B]) = ([q_9, B], [\surd, B], \emptyset)$$

M tìm gặp Blank, nó dừng và chấp nhận chuỗi. Nếu M gặp ký hiệu chưa được đánh dấu khi thành phần thứ 1 là q_8 thì nó dừng và không chấp nhận.

3.4. Dịch qua (Shifting over)

Máy Turing có thể tạo ra một không gian trống trên băng bằng cách dời các ký hiệu không trống trên băng đi sang phải hữu hạn ô. Để làm điều đó đầu đọc phải thực hiện dịch phải, lặp lại việc lưu ký hiệu đọc được vào bộ điều khiển và thay thế chúng bằng ký hiệu đọc được ở ô bên trái. Nếu có đủ ô trống, TM cũng có thể chuyển dịch một khối ký hiệu sang trái một cách tương tự.

Thí dụ 7.6 : Xây dựng TM $M(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ dịch toàn bộ các ký hiệu không trống trên băng sang phải 2 ô.

Ta giả sử không có Blank giữa các ký hiệu không trống, vì vậy khi đầu đọc gặp Blank thì nó đã dịch xong các ký hiệu khác trống trên băng. Tập các trạng thái Q chứa các phần tử dạng $[q, A_1, A_2]$ với $q = q_1$ hoặc q_2 và $A_1, A_2 \in \Gamma$. Gọi X là một ký hiệu đặc biệt được chấp nhận trên băng của M , nó không được sử dụng với mục đích nào khác ngoài quá trình dịch chuyển trên băng. M bắt đầu với trạng thái $[q_1, B, B]$ và hàm chuyển thực hiện như sau:

Với $A_i \in \Gamma - \{B, X\}$

$$1) \quad \delta([q_1, B, B], A_1) = ([q_1, B, A_1], X, R)$$

M lưu ký hiệu đọc đầu tiên vào thành phần thứ 3 trong bộ điều khiển, ghi X vào ô đang đọc rồi dịch sang phải.

$$2) \quad \delta([q_1, B, A_1], A_2) = ([q_1, A_1, A_2], X, R)$$

M chuyển ký hiệu ở thành phần thứ 3 sang thành phần thứ 2, lưu trữ ký hiệu đọc được vào thành phần thứ 3, viết X vào ô đang đọc rồi dịch sang phải.

$$3) \quad \delta([q_1, A_1, A_2], A_3) = ([q_1, A_2, A_3], A_1, R)$$

Bắt đầu từ bước chuyển này, M lần lượt đọc vào một ký hiệu, ghi nó vào thành phần thứ 3, chuyển ký hiệu được ghi trước đó ở thành phần thứ 3 sang thành phần thứ 2, chép lại ký hiệu ở thành phần thứ 2 vào ô đang đọc rồi dịch sang phải.

$$4) \quad \delta([q_1, A_{i-2}, A_{i-1}], A_i) = ([q_1, A_{i-1}, A_i], A_{i-2}, R)$$

$$5) \quad \delta([q_1, A_{n-1}, A_n], B) = ([q_2, A_n, B], A_{n-1}, R)$$

Cho đến khi M gặp B , nó dọc nốt 2 ký hiệu cuối đang giữ trong bộ nhớ để bắt đầu đi vào trạng thái kết thúc.

$$6) \quad \delta([q_2, A_n, B], B) = ([q_2, B, B], A_n, L)$$

Cuối cùng, tất cả các ký hiệu không trống trên băng đã được chuyển dịch sang phải 2 ô. Lúc đó nó sẽ được chuyển sang một trạng thái nào đó (có thể quay về trái, trở về đầu băng) để thực hiện một chức năng khác.

3.5. Chương trình con (Subroutines)

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 quy hoặc có tham số. Ý tưởng chung là ta viết một phần chương trình của TM như là một chương trình con. Nó sẽ được thiết kế 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 TM khác hoặc là một trạng thái nào đó trong một TM khác. Nghĩa là từ trạng thái trở về của TM này ta tiếp tục các phép chuyển của một TM khác, sự kiện này có ý nghĩa như là 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.

Thí dụ 7.7 : Thiết kế TM thực hiện phép nhân 2 số nguyên m, n .

. Input : $0^m 10^n$

. Output : $0^{m \times n}$

M bắt đầu với $0^m 10^n$ trên băng và kết thúc với $0^{m \times n}$ trên băng được bao quanh bởi các Blank.

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 giải thuật trên là thủ tục COPY để chép n số 0 sang phải. Thủ tục này được xác định bằng các hàm chuyển sau:

	0	1	2	B
q₁	(q ₂ , 2, R)	(q ₄ , 1, L)		(q ₃ , 0, L)
q₂	(q ₂ , 0, R)	(q ₂ , 1, R)		
q₃	(q ₃ , 0, L)	(q ₃ , 1, L)	(q ₁ , 2, R)	
q₄		(q ₅ , 1, R)	(q ₄ , 0, L)	

Ở trạng thái q_1 nhìn thấy 0, M đổi 0 thành 2 và đi vào trạng thái q_2 . Ở trạng thái q_2 , M dịch phải tới Blank viết 0 rồi dịch trái trong trạng thái q_3 . Khi ở trạng thái q_3 mà gặp 2, M 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 thái khởi đầu $q_0 0^m 10^n$ thành $B 0^{m-1} 1 q_1 0^n 1$. 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 $10^n 1$ bị xoá và quá trình tính toán sẽ dừng ở trạng thái q_{12} . Các hàm chuyển bổ sung như sau :

	0	1	2	B
q₅	(q ₇ , 0, L)			
q₇		(q ₈ , 1, L)		
q₈	(q ₉ , 0, L)			(q ₁₀ , B, R)
q₉	(q ₉ , 0, L)			(q ₀ , B, R)
q₁₀		(q ₁₁ , B, R)		
q₁₁	(q ₁₁ , B, R)	(q ₁₂ , B, \emptyset)		

IV. CÁC BIẾN DẠNG CỦA MÁY TURING

Sau đây, ta sẽ xét thêm một số dạng khác của máy Turing, chúng có vẻ phức tạp và tinh vi hơn, song thực tế chúng cũng đều tương đương với mô hình TM cơ bản đã định nghĩa ở trên.

4.1. Máy Turing với băng vô hạn 2 chiều

Máy Turing với băng vô hạn hai chiều cũng tương tự như mô hình gốc (TM vô hạn một chiều băng), chỉ khác là băng của nó không có cận trái như mô hình gốc, nghĩa là ta xem như TM có vô hạn Blank ở cả hai đầu băng. Vì thế hàm δ được mở rộng thêm bằng cách xét thêm các trường hợp đặc biệt tại cận trái như sau :

Nếu $\delta(q, X) = (p, Y, L)$ thì $qX\alpha \vdash pBY\alpha$

Nếu $\delta(q, X) = (p, B, R)$ thì $qX\alpha \vdash p\alpha$

ĐỊNH LÝ 7.1 : Nếu L được nhận diện bởi TM với băng vô hạn hai chiều thì L cũng được nhận diện bằng TM vô hạn một chiều băng

Chứng minh

Gọi M_2 là TM với băng vô hạn hai chiều $M_2 (Q_2, \Sigma_2, \Gamma_2, \delta_2, q_2, B, F_2)$ nhận diện L. Ta xây dựng M_1 là TM vô hạn một chiều băng nhận diện L. Băng của M_1 có 2 rãnh:

- Rãnh trên biểu diễn cho băng của M_2 phía phải đầu đọc lúc khởi đầu.
- Rãnh dưới biểu diễn cho băng phía trái đầu đọc lúc khởi đầu theo thứ tự ngược lại.

...	A ₋₅	A ₋₄	A ₋₃	A ₋₂	A ₋₁	A ₀	A ₁	A ₂	A ₃	A ₄	A ₅	...
-----	-----------------	-----------------	-----------------	-----------------	-----------------	----------------	----------------	----------------	----------------	----------------	----------------	-----

(a) - Bảng của M_2

A_0	A_1	A_2	A_3	A_4	A_5	...
\varnothing	A_{-1}	A_{-2}	A_{-3}	A_{-4}	A_{-5}	...

(b) - Bảng của M_1

Hình 7.3 - Bảng nhập của TM M_2 và M_1

M_1 thực hiện các phép chuyển tương tự như M_2 nhưng khi M_2 thực hiện các phép chuyển phải đầu đọc thì M_1 làm việc với rãnh trên, khi M_2 thực hiện các phép chuyển bên trái đầu đọc thì M_1 làm việc ở rãnh dưới

Một cách hình thức $M_1 (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_1, B, F_1)$, trong đó :

Q_1 là tập hợp các đối tượng dạng $[q, U]$ hoặc $[q, D]$, trong đó q là trạng thái trong Q_2 , còn U, D dùng chỉ rằng M_1 đang làm việc với rãnh trên (Up) hay rãnh dưới (Down). Các ký hiệu băng của M_1 (các ký hiệu thuộc Γ_1) có dạng $[X, Y]$ trong đó X, Y thuộc Γ_2 , hơn nữa Y có thể là \varnothing là ký hiệu không có trong Γ_2 dùng để đánh dấu ô trái nhất trên băng của M_1 .

Σ_1 là tập hợp các đối tượng dạng $[a, B]$ trong đó $a \in \Gamma_2$.

$F_1 = \{[q, U], [q, D] \mid q \in F_2\}$.

Hàm chuyển δ_1 có dạng như sau:

1) $\delta_1(q_1, [a, B]) = ([q, U], [X, \varnothing], R)$ nếu $\delta_2(q_2, a) = (q, X, R)$

Nếu M_2 chuyển sang phải trong lần chuyển đầu tiên thì M_1 in \varnothing trên rãnh dưới, ghi nhớ U vào thành phần thứ hai của trạng thái và dịch phải. Thành phần thứ nhất của trạng thái lưu trạng thái của M_2 . M_1 in X (ký hiệu mà M_2 in ra) ở rãnh trên.

2) $\forall a \in \Sigma_2 \cup \{B\}$:

$\delta_1(q_1, [a, B]) = ([q, D], [X, \varnothing], R)$ nếu $\delta_2(q_2, a) = (q, X, L)$

Nếu M_2 chuyển sang trái trong lần chuyển đầu tiên thì M_1 in \varnothing trên rãnh dưới, ghi nhớ D vào thành phần thứ hai của trạng thái và dịch phải. Thành phần thứ nhất của trạng thái lưu trạng thái của M_2 . M_1 in X (ký hiệu mà M_2 in ra) ở rãnh trên.

3) $\forall [X, Y] \in \Gamma_1$, với $Y \neq \varnothing$ và $A = L$ hoặc R :

$\delta_1([q, U], [X, Y]) = ([p, U], [Z, Y], A)$ nếu $\delta_2(q, X) = (p, Z, A)$

M_1 ở rãnh trên thực hiện tương tự như M_2 .

4) $\delta_1([q, D], [X, Y]) = ([p, D], [X, Z], A)$ nếu $\delta_2(q, Y) = (p, Z, \bar{A})$

(Trong đó nếu $A = L$ thì $\bar{A} = R$ và nếu $A = R$ thì $\bar{A} = L$)

Ở rãnh dưới, M_1 làm tương tự M_2 nhưng dịch chuyển đầu đọc theo hướng ngược lại.

5) $\delta_1([q, U], [X, \varnothing]) = \delta_1([q, D], [X, \varnothing]) = ([p, C], [Y, \varnothing], R)$

nếu $\delta_2(q, X) = (p, Y, A)$

(Trong đó $C = U$ nếu $A = R$, $C = D$ nếu $A = L$)

M_1 làm tương tự M_2 ở ô khởi đầu, công việc tiếp theo của M_1 thực hiện ở rãnh trên hay dưới phụ thuộc vào hướng chuyển đầu đọc của M_2 .

4.2. Máy Turing với nhiều băng vô hạn hai chiều

Xét máy Turing có một bộ điều khiển có k đầu đọc và k băng vô hạn hai chiều. Mỗi phép chuyển của máy Turing, phụ thuộc vào trạng thái của bộ điều khiển và ký tự đọc được tại mỗi đầu đọc, nó có thể thực hiện các bước sau :

- 1) Chuyển trạng thái.
- 2) In ký hiệu mới tại mỗi đầu đọc để thay thế ký hiệu vừa đọc.
- 3) Đầu đọc có thể giữ nguyên vị trí hoặc dịch trái hoặc dịch phải 1 ô một cách độc lập nhau.

Khởi đầu input xuất hiện trên băng thứ nhất, các băng khác chỉ toàn Blank.

Một máy Turing như vậy gọi là máy Turing với nhiều băng vô hạn hai chiều.

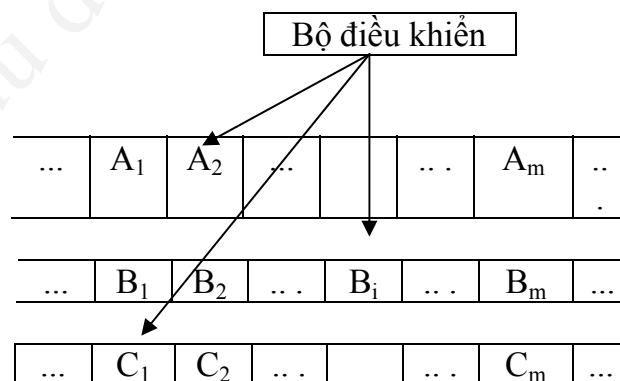
ĐỊNH LÝ 7.2 : Nếu L được nhận dạng bởi máy Turing nhiều băng vô hạn hai chiều thì nó cũng được nhận dạng bởi máy Turing một băng vô hạn hai chiều.

Chứng minh

Giả sử L được nhận diện bởi máy Turing k băng vô hạn hai chiều M_1 , ta xây dựng máy Turing M_2 một băng với $2k$ rãnh, 2 rãnh sẽ mô phỏng một băng của M_1 bằng cách: một rãnh giữ ký hiệu trên băng của M_1 một rãnh kia giữ ký hiệu đánh dấu vị trí đầu đọc.

Mỗi phép chuyển của M_1 được mô phỏng bằng M_2 như sau:

M_2 xuất phát tại vị trí trái nhất chứa ký hiệu đánh dấu đầu đọc, M_2 quét sang phải, khi qua mỗi ô có đánh dấu vị trí đầu đọc nó ghi nhớ ký hiệu tại vị trí này và đếm số vị trí đầu đọc đã gặp. Khi M_2 đi sang phải và đã đếm đủ k đầu đọc thì nó đã có đủ thông tin để xác định phép chuyển tương tự như M_1 , M_2 lại quét từ phải sang trái, khi đi ngang qua mỗi ô có đánh dấu đầu đọc nó in ký hiệu thay thế ký hiệu tại đầu đọc (như M_1) chuyển vị trí đánh dấu đầu đọc (như M_1 chuyển đầu đọc của nó). Cuối cùng M_2 đổi trạng thái trong bộ điều khiển của nó thành trạng thái mà M_1 chuyển tới.



Đầu đọc 1		X				
Băng 1	A ₁	A ₂	A _m
Đầu đọc 2				X		
Băng 2	B ₁	B ₂	...	B _i	...	B _m
Đầu đọc 3	X					

Bảng 3	C_1	C_2	\dots	\dots	C_m
--------	-------	-------	---------	---------	-------

Hình 7.4 - Máy Turing 1 băng mô phỏng máy Turing 3 băng

Thí dụ 7.8 : Ngôn ngữ $\{ww \mid w \in (0+1)^*\}$ có thể được chấp nhận bởi một máy Turing có 2 băng bằng cách như sau: Đầu tiên, nó chép lại chuỗi nhập ở băng thứ nhất lên băng thứ hai. Sau đó, trên băng thứ nhất đầu đọc chuyển dần từ cận trái sang cận phải của chuỗi, trong khi trên băng thứ hai đầu đọc lại chuyển ngược lại từ cận phải sang cận trái của chuỗi đó. Chuỗi được chấp nhận nếu suốt quá trình đó, các ký hiệu đọc được trên 2 băng luôn luôn đồng nhất.

Như ta đã biết, để đoán nhận ngôn ngữ này bằng TM một băng thì đầu đọc phải dịch chuyển tới lui rất nhiều lần để so sánh hai nửa của chuỗi nhập từ cả hai đầu băng. Như vậy, số bước dịch chuyển của nó xấp xỉ bằng bình phương độ dài chuỗi nhập, trong khi TM với 2 băng nhập chỉ cần thực hiện số bước chuyển tỷ lệ với độ dài của chuỗi nhập.

4.3. Máy Turing không đơn định

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, lựa chọn hướng chuyển đầu đọc, và lựa chọn ký hiệu in ra trên băng để thay thế ký hiệu vừa đọc được. Máy Turing trong mô hình gốc còn gọi là máy Turing đơn định.

ĐỊNH LÝ 7.3 : Nếu L được chấp nhận bởi máy Turing không đơn định M_1 thì L cũng được chấp nhận bởi một máy Turing đơn định M_2 nào đó.

Chứng minh

Với một trạng thái và một ký hiệu băng bất kỳ của M_1 , có một số hữu hạn các phép chuyển đến trạng thái kế tiếp, ta có thể đánh số các trạng thái này là 1, 2, ... Gọi r là số lớn nhất của số các cách lựa chọn với một cặp trạng thái và ký hiệu bất kỳ. Ta có, mọi dãy các phép chuyển trạng thái đều được chỉ ra bằng một dãy chứa các số từ 1 đến r . Ngược lại một dãy hữu hạn bất kỳ gồm các số từ 1 đến r có thể biểu diễn cho một dãy các phép chuyển nào đó cũng có thể không. M_2 được thiết kế có ba băng:

Băng 1 chứa input

Băng 2 sinh ra dãy chứa các số từ 1 đến r một cách tự động theo tính chất dãy ngắn sinh ra trước, nếu các dãy cùng độ dài thì nó sinh ra theo thứ tự liệt kê số (numerical order).

Băng 3 dùng chép input trên băng 1 vào để xử lý: với mỗi số sinh ra trên băng 2, M_2 chép input trên băng 1 vào băng 3 và thực hiện các phép chuyển theo dãy số trên băng 2.

Nếu có một chuỗi nào đó trên băng 2 làm cho M_2 đi vào trạng thái kết thúc thì M_2 dừng và chấp nhận input. Nếu không có chuỗi nào như vậy thì M_2 không chấp nhận input. Tất nhiên M_2 chấp nhận input khi và chỉ khi M_1 chấp nhận input.

4.4. Máy Turing nhiều chiều

Máy Turing nhiều chiều gồm một bộ điều khiển hữu hạn, nhưng băng của nó là một mảng k chiều vô hạn về cả 2k phía. Với một số k nào đó, phụ thuộc vào trạng thái và một ký hiệu được đọc, máy thay đổi trạng thái, in một ký hiệu mới tại ô đang đọc và dịch chuyển đầu đọc theo một trong 2k phía.

ĐỊNH LÝ 7.4: Nếu L được chấp nhận bởi máy Turing k chiều M_1 thì L cũng được chấp nhận bởi một máy Turing một chiều M_2 nào đó.

(Phần chứng minh, xem như bài tập)

4.5. Máy Turing nhiều đầu đọc

Máy Turing nhiều đầu đọc có k đầu đọc được đánh số từ 1 đến k với k là một số hữu hạn nào đó, nhưng chỉ có một băng input. Một phép chuyển của máy Turing phụ thuộc vào trạng thái và các ký tự được đọc bởi mỗi đầu băng. Mỗi đầu dịch chuyển một cách độc lập sang trái, sang phải hoặc đứng yên.

ĐỊNH LÝ 7.5 : Nếu L được chấp nhận bởi máy Turing k đầu đọc M_1 thì L cũng được chấp nhận bởi một máy Turing một đầu đọc M_2 nào đó.

(Phần chứng minh, xem như bài tập)

V. GIẢ THUYẾT CHURCH

Giả thuyết rằng khái niệm trực giác “**Hàm tính được**” (computable function) 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. Trong khi chúng ta không thể hy vọng để chứng minh giả thuyết Church cũng như những định nghĩa không hình thức về “sự tính được”, chúng ta có thể cho những dẫn chứng về những khả triển 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 lưu trữ, có vẻ như các hàm đệ quy bộ phận thì 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”. 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 được 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 (Random Access Machine) 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 $0, 1, \dots$, 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 sẽ 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 tác dụng như một kiểu RAM.

Mô phỏng mô hình RAM bởi máy Turing

ĐỊNH LÝ 7.6: Một máy Turing có thể mô phỏng một RAM, với điều kiện là mỗi chỉ thị RAM cũng có thể được mô phỏng bởi một TM.

Chứng minh

Ta sử dụng một TM M nhiều băng để thực hiện quá trình mô phỏng. Một băng của M giữ các từ của RAM được cho bởi các giá trị như dưới đây. Băng có dạng sau :

$$\# 0 \cdot v_0 \# 1 \cdot v_1 \# 10 \cdot v_2 \# \dots \# i \cdot v_i \# \dots$$

trong đó v_i là nội dung băng viết dưới dạng nhị phân của từ thứ i . Tại mỗi thời điểm, sẽ có một số hữu hạn các từ của RAM có thể được dùng và M chỉ cần lưu giữ lại các giá trị cho đến khi có một số lượng từ lớn nhất được sử dụng sau đó.

RAM có một số hữu hạn các thanh ghi số học. M dùng một băng để giữ nội dung của mỗi thanh ghi này, một băng để giữ *số đếm vị trí* (location counter), nơi chứa số thứ tự các từ mà chỉ thị kế tiếp sẽ gọi đến. Và một băng nữa dùng như là *thanh ghi địa chỉ bộ nhớ* (memory address register) trong đó lưu giữ số của từ nhớ.

Giả sử rằng 10 bit đầu tiên của một chỉ thị biểu thị một toán tử chuẩn của máy tính, chẳng hạn như LOAD, STORE, ADD, ... và những bit sau đó xác định địa chỉ của một toán hạng. Trong khi chúng ta sẽ xem xét một cách chi tiết việc cài đặt tất cả các chỉ thị máy tính chuẩn, một ví dụ minh họa sẽ cho thấy điều này rõ ràng hơn. Giả sử băng số đếm vị trí của M giữ số i trong hệ nhị phân. M duyệt băng này đầu tiên từ bên trái và tìm thấy $\# i \cdot$. Nếu một khoảng trống được đếm trước khi tìm thấy $\# i \cdot$, có nghĩa là không có chỉ thị nào trong từ i , vì thế RAM và M ngừng lại. Nếu $\# i \cdot$ được tìm thấy, chuỗi bit theo sau ký hiệu \cdot cho đến ký hiệu $\#$ sau đó sẽ được xem xét. Giả sử 10 bit đầu tiên là mã lệnh của "ADD to register 2" và phần chuỗi bit còn lại là một số j trong hệ nhị phân. M thêm 1 vào i trên băng số đếm vị trí và sao chép j vào băng địa chỉ bộ nhớ. Sau đó, M lại tìm kiếm $\# j \cdot$ trên băng đầu tiên, một lần nữa lại bắt đầu từ bên trái (chú ý rằng $\# 0 \cdot$ đánh dấu vị trí cận trái). Nếu $\# j \cdot$ không tìm thấy, ta giả sử từ j giữ 0 và đi tiếp đến chỉ thị kế tiếp của RAM. Nếu $\# j \cdot v_j \#$ được tìm thấy, v_j sẽ được thêm vào nội dung của thanh ghi 2, nơi chứa chính nó trên băng. Tiếp tục lặp lại vòng lặp này với chỉ thị kế tiếp.

Lưu ý rằng trong giải thuật này, mặc dù mô phỏng RAM dùng một máy Turing nhiều băng, nhưng theo Định lý 7.2, nếu ta dùng TM với một băng vô hạn hai chiều cũng sẽ thành công song sẽ phức tạp hơn.

Vi. MÁY TURING NHƯ LÀ MỘT BỘ LIỆT KÊ

Ta đã xét máy Turing như là một máy dùng nhận dạng ngôn ngữ và tính toán các hàm. Một quan điểm rất có ích nữa là xem máy Turing như là bộ liệt kê, tức là nó có khả năng sinh ra ngôn ngữ.

Xét máy Turing có nhiều băng, một trong các băng đó được xem là băng output, trên băng này một ký hiệu được viết lên sẽ không bị thay đổi và đầu đọc của băng này không bao giờ dịch trái.

Giả sử trên băng output, M sẽ viết chuỗi các ký tự thuộc bộ chữ cái Σ , các chuỗi được viết ngăn cách nhau bởi dấu $\#$. Ta định nghĩa $G(M)$ là ngôn ngữ sinh bởi máy Turing M , là tập hợp tất cả các từ $w \in \Sigma^*$ được viết giữa hai dấu $\#$ trên băng output. Chú ý rằng trừ khi M không dừng, $G(M)$ luôn luôn hữu hạn. Ta cũng không yêu cầu các từ được sinh ra theo một thứ tự nào đó, và cũng không yêu cầu mỗi từ chỉ sinh ra đúng một lần.

6.1. Tính chất đệ qui liệt kê của tập sinh

BỔ ĐỀ 7.1: Nếu L là $G(M_1)$ với TM M_1 nào đó thì L là tập đệ qui liệt kê

Chứng minh

Ta xây dựng M_2 có nhiều hơn M_1 một băng, M_2 sẽ thực hiện tương tự M_1 , M_2 dùng tất cả các thành phần của M_1 chỉ trừ băng input của M_1 , nhưng khi M_1 in $\#$ trên băng output của M_1 thì M_2 lấy input của M_2 so sánh với từ vừa được sinh trên băng output của M_1 . Nếu giống thì M_2 chấp nhận, ngược lại M_2 tiếp tục làm tương tự M_1 . Rõ ràng M_2 chấp nhận input w nếu và chỉ nếu M_1 sinh ra w , vậy $G(M_1)$ là tập đệ qui liệt kê.

Chứng minh điều ngược lại của bổ đề trên là khó khăn hơn. Giả sử M_1 là bộ nhận dạng của một tập đệ qui liệt kê $L \subseteq \Sigma^*$ nào đó. Nếu ta cố gắng thiết kế một bộ sinh ra L có thể sinh mọi từ thuộc Σ^* theo thứ tự nào đó là w_1, w_2, \dots , ta cho chạy M_1 trên w_1 , nếu M_1 chấp nhận thì sinh ra w_1 . Rồi chạy M_1 với w_2, \dots , cứ lần lượt như thế với mọi từ. Phương pháp này chỉ đúng nếu M_1 dừng trên mọi input. Tuy nhiên, do có các tập đệ qui liệt kê nhưng không đệ qui vì vậy M_1 có thể không dừng với một input w_i nào đó và tất nhiên ta sẽ không bao giờ xét được các từ sau đó w_{i+1}, w_{i+2}, \dots ngay cả khi M_1 chấp nhận chúng.

Từ phân tích trên, ta thấy vấn đề đặt ra là phải thiết kế bộ sinh sao cho nó có thể tránh được trường hợp trên. Để làm như vậy trước hết ta đánh số thứ tự các từ thuộc Σ^* rồi ta sinh ra từng cặp số nguyên dương (i, j) . Việc sinh ra cặp (i, j) có ý nghĩa như là M_1 sinh ra từ thứ i bằng j bước. Cụ thể, ta đánh số các từ trong Σ^* theo "**thứ tự chuẩn**" (*canonical order*) như sau: liệt kê các từ theo độ dài, với các từ có cùng độ dài được liệt kê theo thứ tự số, tức là nếu $\Sigma = \{a_0, a_1, \dots, a_{k-1}\}$ thì mỗi từ $w \in \Sigma^*$ coi như là một số trong hệ k -phân. Ta có thể thiết kế TM sinh ra các từ theo thứ tự chuẩn là không khó khăn gì.

Thí dụ 7.9 : Nếu $\Sigma = \{0,1\}$ thì các từ liệt kê theo thứ tự chuẩn là: $\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \dots$ Xét cách sinh ra cặp sinh (i, j) sau một lượng thời gian hữu hạn. Nếu sinh theo thứ tự $(1, 1), (1, 2), \dots$ thì sẽ không bao giờ sinh được cặp (i, j) với $i > 1$. Thay vào đó ta cho sinh ra cặp (i, j) theo thứ tự $i + j$, vì số lượng cặp (i, j) thỏa $i + j$ bằng hằng số là hữu hạn nên cặp (i, j) bất kỳ sẽ được sinh ra sau một lượng thời gian hữu hạn, cụ thể ta có thể chứng minh: cặp (i, j) sẽ được sinh ở lần sinh thứ :

$$(i + j - 1)(i + j - 2) / 2 + i.$$

Máy Turing sinh ra các cặp sinh (i, j) viết trong hệ nhị phân là dễ dàng được thiết kế và ta gọi máy Turing này là bộ sinh cặp.

ĐỊNH LÝ 7.7 : Một ngôn ngữ là tập đệ qui liệt kê nếu và chỉ nếu nó là $G(M_2)$ với TM M_2 nào đó.

Chứng minh

Với bổ đề 1 ta chỉ cần chỉ ra rằng nếu $L = L(M_1)$ thì L được sinh ra bởi TM M_2 . M_2 tương tự như bộ sinh cặp. Khi (i, j) được sinh ra, M_2 sản xuất từ thứ i theo thứ tự chuẩn và làm tương tự M_1 trên từ w_i với j bước. Nếu M_1 chấp nhận từ thứ i với j bước thì M_2 sinh ra w_i . Chắc chắn rằng M_2 không sinh ra từ không thuộc L , đặt w là từ thứ i trong thứ tự chuẩn trên bộ chữ cái L và M_1 chấp nhận w bằng j bước. Vì chỉ cần một lượng thời gian hữu hạn để M_2 sinh ra bất kỳ từ nào và làm tương tự như M_1 nên M_2 chắc chắn sinh ra (i, j) . Lúc này w sẽ được sinh ra bởi M_2 . Vậy $L = G(M_2)$

HỆ QUẢ : Nếu L là tập đệ qui liệt kê thì có một bộ sinh sinh ra mỗi từ trong L đúng một lần.

6.2. Tính chất đệ qui của tập sinh

BỔ ĐỀ 7.2: Nếu L là tập đệ qui thì có một bộ sinh in ra các từ của L theo thứ tự chuẩn và không in ra các từ khác.

Chứng minh

Đặt $L = L(M_1) \in \Sigma^*$ trong đó M_1 dừng với mọi input. Ta xây dựng M_2 sinh ra L như sau M_2 sinh ra các từ thuộc Σ^* mỗi lần một từ theo thứ tự chuẩn. Sau khi sinh

ra một từ w , M_2 làm tương tự M_1 trên w . Nếu M_1 chấp nhận w thì M_2 sinh ra w . Vì M_1 chắc chắn dừng nên M_2 cũng sẽ dừng sau hữu hạn bước và chắc chắn sẽ xét mỗi từ thuộc Σ^* . Vậy M_2 sinh ra L theo thứ tự chuẩn.

Điều ngược lại của bổ đề trên cũng đúng, tức là, nếu L được sinh ra theo thứ tự chuẩn thì L là tập đệ quy. Nghĩa là có TM nhận diện M tồn tại, tuy nhiên không có một giải thuật cụ thể cho TM này.

Giả sử M_1 sinh ra L theo thứ tự chuẩn. Một ý nghĩ tự nhiên là ta xây dựng M_2 làm tương tự M_1 trên input w cho tới khi M_1 sinh ra w hoặc sinh ra từ sau w (theo thứ tự chuẩn). Trong trường hợp đầu, M_2 chấp nhận w , trong trường hợp sau M_2 dừng nhưng không chấp nhận w . Rõ ràng nếu L hữu hạn thì M_1 có thể không dừng sau khi sinh ra từ cuối cùng trong L (vì theo định lý trên M_1 không sinh từ nào không thuộc L). Trong trường hợp này M_2 cũng không dừng. Điều này chỉ gặp khi L hữu hạn, nhưng do không có cách xác định TM có sinh ra tập hữu hạn hay không hoặc nếu biết TM sinh ra tập hữu hạn thì cũng không thể biết tập hợp đó là gì. Vậy ta biết là có TM chấp nhận L , nhưng không thể đưa ra một giải thuật cụ thể cho TM này.

ĐỊNH LÝ 7.8 : L là tập đệ quy nếu và chỉ nếu L được sinh ra theo thứ tự chuẩn.

Chứng minh

Ta chỉ cần chứng minh phần nếu.

Khi L hữu hạn thì sẽ có một ô tô-mát chấp nhận L và vì vậy L được chấp nhận bởi TM luôn luôn dừng trên tất cả các input.

VII. SỰ TƯƠNG ĐƯƠNG GIỮA VĂN PHẠM KIỂU 0 VÀ MÁY TURING

Họ văn phạm rộng lớn nhất theo sự phân cấp của Noam Chomsky đòi hỏi các luật sinh có dạng $\alpha \rightarrow \beta$, với α, β là các chuỗi tùy ý chứa ký hiệu văn phạm sao cho $\alpha \neq \varepsilon$. Lớp văn phạm này được biết như là văn phạm *kiểu 0*, văn phạm *ngữ cấu* hay văn phạm *không hạn chế*.

Thí dụ 7.10 : Cho một văn phạm không hạn chế sinh ra ngôn ngữ

$L = \{ a^i \mid i \text{ là lũy thừa dương của } 2 \}$ với tập luật sinh như sau :

$G (\{S, A, B, C, D, E\}, \{a, \varepsilon\}, P, S)$

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. $AE \rightarrow \varepsilon$

$\}$

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 luật sinh (2). Khi C chạm đến cận phải B, nó sẽ thay thế thành D hay E theo luật sinh (3) hoặc (4). Nếu D được chọn thay thế thì D lại quay về trái theo luật sinh (5), cho đến khi gặp cận trái A thì thay thế lại thành C theo luật sinh (6) và cho phép lặp lại chu trình. Còn nếu E được chọn để thay thế, thì theo luật sinh (4), B sẽ biến mất, sau đó E quay về trái theo luật sinh (7) cho đến khi gặp cận trái A thì xóa A và mất đi theo luật sinh (8), cho ra chuỗi có dạng 2^i ký hiệu a, với $i > 0$.

Có thể chứng minh bằng quy nạp theo số bước dẫn xuất rằng nếu luật sinh (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 dương của 2.
- (iii) Aa^iDa^jB , với $i + j$ là một lũy thừa dương của 2.

Khi luật sinh (4) được áp dụng thì ta sẽ có chuỗi dạng Aa^iE , trong đó i là một lũy thừa dương của 2. Sau đó, ta chỉ có thể áp dụng i lần luật sinh (7) để đi tới dạng câu AEa^i . Cuối cùng, với luật sinh (8), ta thu được chuỗi dạng a^i với i là lũy thừa dương của 2.

Phần tiếp theo dưới đây, 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 liệt kê r.e – lớp ngôn ngữ được chấp nhận bởi một máy Turing. Định lý đầu tiên sẽ chứng tỏ rằng mọi ngôn ngữ kiểu 0 phát sinh một tập r.e. Và sau đó ta sẽ xây dựng một giải thuật để liệt kê tất cả các chuỗi thuộc văn phạm kiểu 0.

ĐỊNH LÝ 7.9 : Nếu L là $L(G)$ với một văn phạm không hạn chế $G(V, T, P, S)$ thì L là ngôn ngữ đệ quy liệt kê.

Chứng minh

Thiết lập một máy Turing M không đơn định, hai băng chấp nhận ngôn ngữ L . Băng thứ nhất (băng 1) của TM chứa chuỗi nhập 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, TM lặp lại quá trình sau :

(i) Chọn một cách ngẫu nhiên một vị trí i trên α với $1 \leq i \leq |\alpha|$, nghĩa là TM 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 luật sinh $\beta \rightarrow \gamma$ trong số các luật sinh thuộc tập luật sinh của G .

(iii) Nếu chuỗi con β xuất hiện trong α kể từ vị trí thứ i , TM thay thế chuỗi β bởi γ (dĩ nhiên nếu $|\beta| \neq |\gamma|$ 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 TM 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).

$$\text{Vậy } L(M) = L(G) = L.$$

ĐỊNH LÝ 7.10 : Nếu L là ngôn ngữ đệ quy liệt kê thì $L = L(G)$ với một văn phạm không hạn chế G nào đó.

Chứng minh

Giả sử ngôn ngữ L được chấp nhận bởi máy Turing $M (Q, \Sigma, \Gamma, \delta, q_0, B, F)$. 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 TM.

(ii) G mô phỏng lại quá trình làm việc của của TM trên chuỗi w , bằng cách lặp lại đúng quá trình làm việc của TM.

(iii) Khi bước (ii) kết thúc, với sự xuất hiện của một trạng thái kết thúc $q \in F$ của TM (nghĩa là chuỗi w đã được TM 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 $G (V, \Sigma, P, S_1)$

Với $V = ((\Sigma \cup \{ \varepsilon \}) \times \Gamma) \cup \{ S_1, S_2, \# \}$

Và tập luật sinh P được xây dựng như sau :

1. a) $S_1 \rightarrow \#q_0 S_2\#$

b) $S_2 \rightarrow [a, a] S_2\#, \forall a \in \Sigma$

c) $S_2 \rightarrow \varepsilon$

- Nếu $\delta(q, X) = (p, Y, R)$ với $p, q \in \Sigma; X, Y \in \Gamma$ thì thêm các luật sinh dạng (2a) và (2b) sau đây vào tập luật sinh P :

2. a) $q[a, X][b, Z] \rightarrow [a, Y]p[b, Z], \forall a, b \in \Sigma \cup \{ \varepsilon \}$ và $\forall Z \in \Gamma$

b) $q[a, X]\# \rightarrow [a, Y]p[\varepsilon, B], \forall a \in \Sigma \cup \{ \varepsilon \}$

- Nếu $\delta(q, X) = (p, Y, L)$ với $p, q \in \Sigma; X, Y \in \Gamma$ thì thêm các luật sinh dạng (2c) sau đây vào tập luật sinh P :

c) $[b, Z]q[a, X] \rightarrow q[b, Z]p[a, Y], \forall a, b \in \Sigma \cup \{ \varepsilon \}$ và $\forall Z \in \Gamma$

- Nếu $q \in F$ thì thêm các luật sinh (3a-e) sau đây vào tập luật sinh P :

3. a) $[a, X]q \rightarrow qap, \forall a \in \Sigma \cup \{ \varepsilon \}$ và $\forall X \in \Gamma$

b) $q[a, X] \rightarrow qap, \forall a \in \Sigma \cup \{ \varepsilon \}$ và $\forall X \in \Gamma$

c) $q\# \rightarrow \varepsilon$

d) $\#q \rightarrow \varepsilon$

e) $q \rightarrow \varepsilon$

Dùng các luật sinh (1a-c), ta có chuỗi dẫn xuất :

$$S_1 \Rightarrow_G^* \#q_0 [a_1, a_1][a_2, a_2] \dots [a_n, a_n]\#$$

Chuỗi dẫn xuất này thể hiện hình thái bắt đầu của TM là : $\#q_0 a_1 a_2 \dots a_n \#$. Bắt đầu từ bước này các quy tắc (2a-c) được áp dụng. Lưu ý rằng các luật sinh này trong

G phản ánh các quy tắc chuyển trạng thái đã được thiết kế cho TM. Cho nên quá trình dẫn xuất lại trong G sẽ mô phỏng lại các bước chuyển hình thái trong quá trình làm việc của TM. Nếu quá trình đó chuyển đến một trong những trạng thái kết thúc $q \in F$, tương ứng với trường hợp TM chấp nhận chuỗi $a_1a_2 \dots a_n$, thì trong văn phạm G các quy tắc (3a-e) sẽ được áp dụng tiếp theo và cho phép G dẫn xuất ra chính chuỗi nhập $a_1a_2 \dots a_n$. Hay ta có : $S \Rightarrow_G^* a_1a_2 \dots a_n$

Phần chứng minh $L(M) \subseteq L(G)$ và $L(G) \subseteq L(M)$ xem như bài tập.

Tổng kết chương VII: Với sự giới thiệu mô hình máy Turing như là một mô hình của sự tính toán, người ta còn đi tới khái niệm về độ phức tạp của tính toán hay “độ khó” của các bài toán. Nghiên cứu về độ phức tạp của tính toán là một hướng nghiên cứu hiện đại trong Tin học, nó có ý nghĩa lớn lao về lý thuyết cũng như thực hành. Kết thúc chương này, sự phân lớp ngôn ngữ theo nguyên tắc của Noam Chomsky đã được thể hiện tương đối rõ ràng.

BÀI TẬP CHƯƠNG VII

7.1. Thiết kế máy Turing nhận diện ngôn ngữ:

- a) $\{ 0^n 1^n 0^n \mid n \geq 1 \}$
- b) $\{ ww^R \mid w \in (0+1)^* \}$
- c) Tập hợp các chuỗi chứa 0 và 1, có số số 0 và số số 1 bằng nhau.

7.2. Thiết kế máy Turing tính các hàm số nguyên:

- a) $f(n) = n^2$
- b) $f(n) = 2^n$
- c) $f(n) = n !$

7.3. Xây dựng văn phạm không hạn chế (loại 0) sinh ra các ngôn ngữ sau:

- a) $\{ ww \mid w \in (0+1)^* \}$
- b) $\{ 0^k \mid k = i^2 \text{ và } i \geq 1 \}$
- c) $\{ 0^i \mid i \text{ không là số nguyên tố} \}$

BÀI TẬP LẬP TRÌNH

7.4. Viết chương trình máy tính mô phỏng hoạt động của các TM thiết kế trong bài tập 7.1 và 7.2.

cuu duong than cong . com