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

Lý thuyết Ôtômat và Ngôn ngữ hình thức
MÁY TURING VÀ ỨNG DỤNG

Giáo viên hướng dẫn: NGÔ THỊ HIỀN

Nhóm Sinh viên thực hiện:

NGUYỄN ĐỨC TRUNG

PHẠM NHẬT DUY

Lớp: KSTN Toán Tin K60

HÀ NỘI - 12/2018

Mục lục

Chương 1

Giới thiệu

Ở báo cáo này,chúng ta sẽ xét một loại máy trừu tượng - 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.

Trong bài báo cáo này, nhóm sẽ trình bày về lý thuyết máy Turing và các ứng dụng liên quan.Báo cáo được trình bày theo 3 chương chính:

- Chương 1: Kiến thức cơ bản
- Chương 2: Lý thuyết máy Turing
- Chương 3: Các ứng dụng liên quan đến máy Turing

Về phần công việc, nhóm em chia như sau:

- Bạn Nguyễn Đức Trung: Máy Turing đa định, máy Turing nhiều dải băng và thuật toán đưa chúng về máy Turing đơn định 1 dải băng
- Bnaj Phạm Nhật Duy: Máy Turing đơn định một dải băng và các ứng dụng liên quan.

Chương 2

Kiến thức cơ sở

2.1 Xâu

- Xâu là một dãy hữu hạn các kí tự
- Bảng chữ cái là một tập hợp các kí tự hữu hạn
- Độ dài xâu là số các kí tự trong xâu đó
- Xâu rỗng:ϵ, độ dài xâu rồng bằng 0
- Các phép toán trên xâu:

    1. Ghép nối xâu

       Cho 2 xâu:$a = a_1a_2...a_n, b = b_1b_2...b_m$ trên bảng chữ cái A. Ghép nối 2 xâu trên ta được một xâu mới $c = ab = a_11_2...a_nb_1b_2...b_m$

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

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

  $s\epsilon = \epsilon s = s$

- Phép ghép nối có tính chất kết hợp

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

- Kí hiệu w^n, với n là số tự nhiên

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

2. Xâu con:

   s là xâu con của w nếu $\exists s_0, s_1$ sao cho $s_0 s s_1 = w$

   $s_0 = \epsilon$ thì s được gọi là prefix của w

   $s_1 = \epsilon$ thì s được gọi là suffix của w

3. Phép đảo ngược xâu

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

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

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

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

2.2 Ngôn ngữ

- Ngôn ngữ là tập các xâu trên bảng chữ cái
- Các phép toán trên ngôn ngữ

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

  1. Phép hợp

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

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

     Định nghĩa phép hợp có thể mở rộng cho hữu hạn các ngôn ngữ:

     $$\bigcup_{i=1}^{n} = \left\{w \in A^* | w \in L_i, i = \overline{1,n}\right\}$$

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

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

  2. Phép giao

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

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

     Định nghĩa phép giao có thể mở rộng cho hữu hạn các ngôn ngữ:

$$\bigcap_{i=1}^{n} = \left\{w \in A^* | w \in L_i, i = \overline{1,n}\right\}$$

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

- Tính chất giao hoán: $L_1 \cap L_2 = L_2 \cap L_1$
- Tính chất kết hợp: $(L_1 \cap L_2) \cap L_3 = L_1 \cap (L_2 \cap L_3)$
- $\forall L_1 : L_1 \cap \varnothing = \varnothing \cup L_1 = \varnothing$ và $L_1 \cap A^* = L_1$
- Tính chất phân phối đối với phép hợp và phép giao:

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

3. Phép nhân ghép

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

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

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

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

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

- Phép lấy phần bù Ngôn ngữ phần bù của ngôn ngữ L trên bảng chữ cái A, kí hiệu C_AL, là một ngôn ngữ trên bảng chữ cái A

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

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

- $C_A\{\epsilon\} = A^+, C_AA^+ = \{\epsilon\}$
- $C_A\varnothing = A^*, C_AA^* = \varnothing$
- $C_A\left(C_AL_1 \cup C_AL_2\right) = L_1 \cap L_2$

2.3 Automat đơn định(Determinictic Finite Automata)

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

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

   $\delta(q, a) = p$

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

2. Định nghĩa

   M là một DFA:

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

   Trong đó:

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

Chương 3

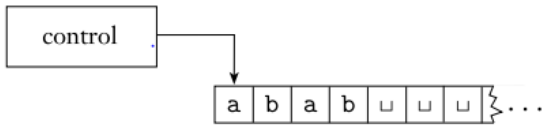
Máy Turing

3.1 Khái niệm về máy Turing

Trước khi đưa ra định nghĩa chính xác, ta bắt đầu với một mô tả sơ lược về máy Turing. Máy Turing sử dụng một dải băng (tape) (đôi khi, ta nói tắt là "dải" hoặc "băng") dài vô hạn mà được chia thành các ô vuông con, mỗi ô vuông chứa một ký tự. Dải băng này bị chặn ở phía trái và vô hạn về bên phải. Do đó, khi nói đến đầu dải băng thì ta mặc định đếm từ bên trái. Bên cạnh đó, máy có một đầu đọc/ghi (tape head) có thể đọc, ghi (read, write) ký tự (symbol) lên dải băng, và di chuyển qua lại trên dải băng. Không có sự nhầm lẫn gì, đôi khi, ta sẽ sử dụng thuật ngữ "đầu đọc" thay cho "đầu đọc/ghi".

Ban đầu, dải băng chỉ chứa xâu đầu vào với độ dài hữu hạn (input string) và trống ở phần còn lại. Nếu máy cần lưu thông tin, nó có thể ghi thông tin này vào dải băng. Để đọc thông tin, máy có thể di chuyển đầu đọc của nó

đến nơi cần đọc. Máy tiếp tục tính toán (computing) đến tận khi nó quyết định trả ra kết quả. Các kết quả có thể là "chấp nhận" (accept) hoặc "từ chối" (reject), nhận được khi máy ở vào trạng thái chấp nhận (accepting state) hoặc từ chối (rejecting state) tương ứng. Nếu nó không bao giờ ở vào các trạng thái đó, nó sẽ chạy mãi không dừng (halt).



Hình 3.1: Giản đồ của một máy Turing

Tiếp theo, ta điểm qua một chút sự khác nhau giữa máy Turing và otomat hữu hạn.

1. Một máy Turing có thể ghi lên dải băng và đọc từ nó. (DFA chỉ đọc)

2. Đầu đọc-ghi có thể di chuyển cả về bên phải lẫn bên trái của dải băng.(DFA: đầu đọc chỉ di chuyển về một phía (bên phải))

3. Máy Turing có thể không dừng. (DFA luôn trả ra kết quả "chấp nhận" hoặc "từ chối" sau hữu hạn bước)

Ta xét ví dụ thiết kế một máy Turing M_1 để kiểm tra một xâu có thuộc ngôn ngữ $B = \{w\#w|w \in \{0,1\}^*\}$ hay không. Ta muốn M_1 chấp nhận nếu đầu vào của nó thuộc B và từ chối nếu ngược lại. Đầu đọc/ghi sẽ dịch chuyển qua lại giữa hai phần chia cách bởi dấu # của xâu đầu vào. Mỗi

lần dịch chuyển, nó sẽ kiểm tra một ký tự bên trái và ký tự tương ứng bên phải (tức là, hai ký tự có cùng khoảng cách đến dấu #) có giống nhau hay không. Các ký tự đã được kiểm tra thì sẽ được M_1 gạch đi. Nếu nó gạch tất cả các kí tự, điều này có nghĩa là xâu đầu vào thuộc B, và M_1 đi đến trạng thái chấp nhận. Nếu nó tìm ra hai ký tự ở các vị trí tương ứng không giống nhau, nó đi đến trạng thái từ chối. Thuật toán ứng với M_1 sẽ như sau.

M_1 : "Ứng với xâu đầu vào w:

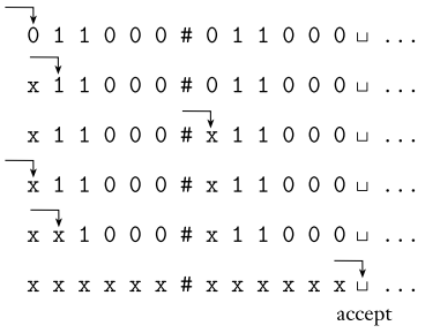
1. Di chuyển qua lại trên dải băng tới các vị trí tương ứng ở hai phía của ký tự # để kiểm tra liệu các vị trí tương ứng chứa cùng kí tự. Nếu kiểm tra trên trả kết quả là sai hoặc xâu đầu vào không chứa #, **từ chối**. Ngược lại, gạch các kí tự đã được kiểm tra.

2. Khi tất cả các kí tự bên trái dấu # bị gạch, kiểm tra các kí tự chưa bị gạch của xâu đầu vào bên phải dấu #. Nếu còn kí tự như vậy chưa bị gạch , **từ chối**; ngược lại, **chấp nhận**."

Hình 3.2 minh họa quá trình chạy thuật toán với xâu 011000#011000.

Tiếp theo, ta đưa ra một định nghĩa chính xác cho máy Turing.

Trọng tâm trong định nghĩa máy Turing là hàm chuyển δ bởi vì nó cho ta biết cách máy đi từ trạng thái này tới trạng thái tiếp theo. Đối với một máy Turing, δ có dạng : $Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$. Nghĩa là, khi máy ở trong một trạng thái q nào đó, đầu đọc tại ô vuông trên dải băng chứa ký tự a, giả thiết $\delta(q, a) = (r, b, L)$, thì máy viết ký tự b thay thế a tại ô vuông đó,



Hình 3.2: Giản đồ của một máy Turing

và đi đến trạng thái r. Thành phần thứ ba nhận giá trị L hoặc R, tương ứng xác định liệu đầu đọc di chuyển sang trái hay sang phải sau khi viết. Trong trường hợp vừa xét, L ứng với bên trái.

Chúng ta mô tả máy Turing bởi bộ 7 thành phần:

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

trong đó:

Q : tập hợp, phần tử của nó được gọi là trạng thái,

Σ : bảng chữ cái ứng với xâu đầu vào và không chứa **ký tự trống** $\sqcup$

Γ : bảng chữ cái bao gồm các ký tự được phép viết lên dải băng,
ta gọi các kí tự này là **ký tự dải**, và $\sqcup \in \Gamma$, $\Sigma \subset \Gamma$

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ là hàm chuyển,

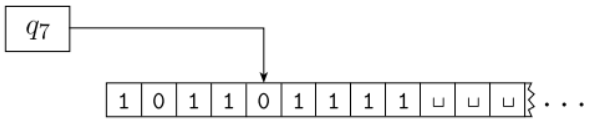
$q_0 \in Q$ là trạng thái bắt đầu,

$q_a \in Q$ là trạng thái **chấp nhận**,

$q_r \in Q$ là trạng thái **từ chối**, và $q_r \neq q_a$.

Một máy Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ thực hiện tính toán (compute) như sau. Ban đầu, M nhận xâu đầu vào $w = w_1 w_2 w_3 \dots w_n \in \Sigma^*$ trên n ô vuông đầu tiên của dải băng, và phần còn lại của dải băng được điền các kí tự trống, đầu đọc/ghi đặt ở ô đầu tiên của dải băng. Chú ý rằng, Σ không bao gồm kí tự trống, do đó, kí tự trống đầu tiên xuất hiện trên dải băng sẽ đánh dấu vị trí kết thúc của đầu vào. Quá trình tính toán diễn ra theo luật được mô tả bởi hàm chuyển. Nếu M cố gắng chuyển đầu đọc về bên trái khi nó đang đứng ở ô đầu tiên(tức là, ô "trái nhất") của dải băng, đầu đọc sẽ vẫn giữ nguyên vị trí của nó mặc dù hàm chuyển xác định giá trị di chuyển là "L". Quá trình tính toán tiếp tục cho đến khi máy đi đến trạng thái chấp nhận hoặc từ chối, tức là trạng thái dừng. Nếu điều đó không xảy ra, máy sẽ chạy mãi.

Trong quá trình máy Turing tính toán, sự thay đổi sẽ xảy ra ở trạng thái hiện tại của máy, nội dung hiện tại trên dải băng, vị trí hiện tại của đầu đọc. Mỗi giá trị của bộ ba trên được gọi là một **cấu hình** (configuration) của máy Turing. Ta biểu diễn cấu hình theo cách sau. Với một trạng thái q, và hai xâu u, v thuộc Γ^*, ta viết uqv biểu thị cấu hình mà *trạng thái hiện tại là q, nội dung hiện tại trên dải băng là uv, và vị trí hiện tại của đầu đọc là ở kí tự đầu tiên của v*. Bên phải vị trí cuối cùng của v, chỉ chứa kí tự trống. Chẳng hạn, $1011q_701111$ biểu thị cấu hình máy đang ở trạng thái q_7, xâu trên dải băng là 101101111, vị trí hiện tại của đầu đọc là số 0 thứ hai. Hình 3.3 miêu tả máy Turing với cấu hình đó. Tiếp theo, ta nói rằng cấu hình C_1 **dẫn đến** cấu hình C_2 nếu máy Turing có thể chuyển từ C_1 sang C_2 sau một bước. Cụ thể như sau:



Hình 3.3: Máy Turing với cấu hình $1011q_701111$. Nguồn [1]

Giả thiết, ta có $a, b, c \in \Gamma$, $u, v \in \Gamma^*$, và các trạng thái q_i, q_j. Trong trường hợp đó, uaq_ibv và uq_jacv là hai cấu hình. Nói rằng:

$$uaq_ibv \text{ dẫn đến } uq_jacv$$

nếu $\delta(q_i, b) = (q_j, c, L)$. Đây ứng với trường hợp máy Turing di chuyển sang trái. Tương tự cho trường hợp di chuyển sang phải, ta nói

$$uaq_ibv \text{ dẫn đến } uacq_jv$$

nếu $\delta(q_i, b) = (q_j, c, R)$.

Các trường hợp đặc biệt xảy ra khi đầu đọc ở một trong các vị trí đầu mút(tức vị trí trái nhất hoặc phải nhất) của cấu hình. Với đầu mút trái, cấu hình q_ibv dẫn đến q_jcv nếu di chuyển sang trái(vì ta ngăn máy di chuyển sang trái từ ô đầu tiên của dải băng), và nó dẫn đến cq_jv với bước di chuyển sang phải. Với mút phải, cấu hình uaq_i tương đương với $uaq_i\sqcup$ bởi vì chúng ta giả thiết rằng các kí tự trống lấp đầy phần còn lại của dải băng. Do đó, chúng ta có thể xử lý trường hợp này như trường hợp sang phải thông thường (với $b = \sqcup$).

Cấu hình bắt đầu (start configuration) của M với xâu đầu vào w là cấu hình q_0w, chỉ ra rằng máy ở trạng thái bắt đầu q_0 với đầu đọc ở ô đầu tiên (ô trái nhất) trên dải băng. Ở một **cấu hình chấp nhận** (accepting

configuration), trạng thái của cấu hình là q_a. Ở một **cấu hình từ chối** (rejecting configuration), trạng thái của cấu hình là q_r. Các cấu hình chấp nhận và từ chối được gọi là các **cấu hình dừng** (halting configuration) vì không dẫn đến cấu hình nào nữa (máy dừng khi nó gặp các trạng thái dừng).

Một máy Turing **chấp nhận** xâu đầu vào w nếu tồn tại một dãy cấu hình $C_1, C_2, \ldots, C_n$, trong đó:

1. C_1 là cấu hình bắt đầu của M với xâu đầu vào w,
2. C_i dẫn đến C_{i+1}, và
3. C_k là cấu hình chấp nhận.

Tập các xâu được chấp nhận bởi M được gọi là **ngôn ngữ của** M, hoặc **ngôn ngữ được đoán nhận bởi M**, ký hiệu là $L(M)$.

Ta nói một ngôn ngữ có tính chất **Turing-đoán nhận được** (Turing-recognizable) nếu có một máy Turing đoán nhận nó. Khi máy Turing bắt đầu chạy với một xâu đầu vào, ba kết quả có thể xảy ra. Hoặc máy chấp nhận hoặc máy từ chối hoặc máy chạy mãi (mà sau đây, ta sẽ gọi đó là vòng lặp (loop)).

Một máy Turing M có thể không chấp nhận một xâu đầu vào w bởi vì nó rơi vào trạng thái q_r hoặc vì nó rơi vào vòng lặp. Đôi khi, phân biệt một máy đang rơi vào vòng lặp với một máy chỉ đơn thuần chạy lâu (tốn thời gian) là khó khăn. Vì lý do này, ta muốn tìm kiếm những máy Turing luôn đi đến một cấu hình dừng cho mọi xâu đầu vào. Một máy như vậy được gọi

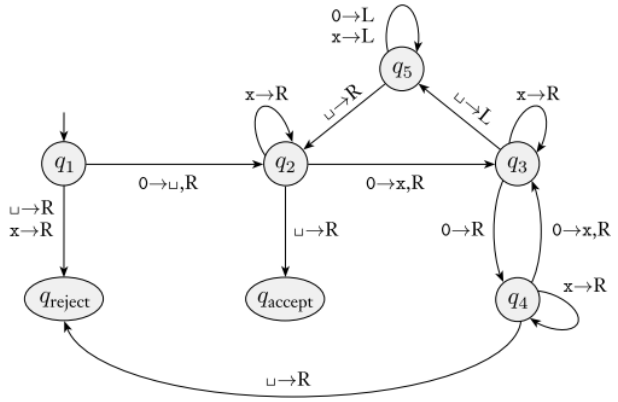
là **máy quyết định** (decider) vì với mỗi xâu đầu vào, chúng luôn dẫn đến quyết định chấp nhận hoặc từ chối. Ta nói một ngôn ngữ là **Turing-quyết định được** (Turing-decidable) nếu có một máy quyết định đoán nhận nó. Tiếp theo, ta đưa ra các ví dụ về ngôn ngữ Turing-quyết định được. Ta mô tả máy quyết định M_2 đoán nhận ngôn ngữ $A = \{0^{2^n} | n \geq 0\}$, bao gồm tất cả các xâu được thành lập từ ký tự 0 và có độ dài là lũy thừa của 2.

M_2 : "Với xâu đầu vào w:

1. Quét từ trái qua phải dải băng, luân phiên gạch các số 0.
2. Nếu ở bước 1, băng chỉ chứa một số 0, **chấp nhận**.
3. Nếu ở bước 1, băng chứa nhiều hơn một số 0 và số lượng số 0 là lẻ, **từ chối**.
4. Chuyển đầu đọc về mút trái của dải băng.
5. Quay lại bước 1."

Mỗi lần lặp lại bước 1, máy gạch đi một nửa số lượng số 0 hiện có. Khi máy quét qua hết dải băng ở bước 1, nó biết số lượng số 0 đã gạch là chẵn hay lẻ. Nếu số đó là lẻ và lớn hơn 1, số lượng số 0 ban đầu ở xâu đầu vào không là lũy thừa của 2. Do đó, máy từ chối. Tuy nhiên, nếu số số 0 đã gặp là 1, số số 0 ban đầu phải là lũy thừa của 2 (là 1 hoặc 2). Do đó, máy chấp nhận.
Bây giờ ta đưa ra một miêu tả hình thức cho $M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_a, q_r)$

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_a, q_r\}$,
- $\Sigma = \{0\}$,
- $\Gamma = \{0, \times, \sqcup\}$ ($\times$ là dấu gạch chéo).
- Ta mô tả δ ở sơ đồ dưới đây (xem hình 3.4).
- Các trạng thái bắt đầu, chấp nhận và từ chối tương ứng là q_1, q_a, q_r.



Hình 3.4: Sơ đồ trạng thái cho máy quyết định M_2. Nguồn [1]

Trong sơ đồ trạng thái trên, nhãn $0 \to \sqcup, R$ xuất hiện trên bước chuyển từ q_1 sang q_2. Nhãn này biểu thị rằng khi ở trạng thái q_1 với đầu đọc đang đọc 0, máy chuyển sang trạng thái q_2, viết $\sqcup$, và chuyển đầu đọc sang bên phải. Nói cách khác, $\delta(q_1, 0) = (q_2, \sqcup, R)$. Chúng ta viết ngắn gọn $0 \to R$ trong bước chuyển từ q_3 sang q_4, để biểu thị máy chuyển sang phải khi đọc 0 ở trạng thái q_3 nhưng không thay đổi kí tự trên băng, do đó $\delta(q_3, 0) = (q_4, 0, R)$.

Máy này bắt đầu bằng việc viết một ký tự trống lên số 0 đầu tiên trên dải

băng để mà nó có thể tìm mút bên trái của dải băng ở bước 4.

Tiếp theo ta đưa ra một ví dụ chạy mẫu của máy trên với xâu đầu vào là 0000. Cấu hình bắt đầu là $q_1 0000$. Dãy cấu hình của máy được xác định

$q_1 0000$
$\sqcup q_2 000$
$\sqcup x q_3 00$
$\sqcup x 0 q_4 0$
$\sqcup x 0 x q_3 \sqcup$
$\sqcup x 0 q_5 x \sqcup$
$\sqcup x q_5 0 x \sqcup$
$\sqcup q_5 x 0 x \sqcup$
$q_5 \sqcup x 0 x \sqcup$
$\sqcup q_2 x 0 x \sqcup$
$\sqcup x q_2 0 x \sqcup$
$\sqcup x x q_3 x \sqcup$
$\sqcup x x x q_3 \sqcup$
$\sqcup x x q_5 x \sqcup$
$\sqcup x q_5 x x \sqcup$
$\sqcup q_5 x x x \sqcup$
$q_5 \sqcup x x x \sqcup$
$\sqcup q_2 x x x \sqcup$
$\sqcup x q_2 x x \sqcup$
$\sqcup x x q_2 x \sqcup$
$\sqcup x x x q_2 \sqcup$
$\sqcup x x x \sqcup q_{\text{accept}}$

như dưới đây (chú ý là ta sẽ đọc trong từng cột theo thứ tự từ trên xuống dưới, các cột được sắp từ trái qua phải).

3.2 Các biến thể của máy Turing

Có nhiều định nghĩa khác cho máy Turing, bao gồm các phiên bản cho máy Turing nhiều dải băng hoặc máy Turing không đơn định. Chúng được gọi là **các biến thể** (variant) của mô hình máy Turing. Mô hình ban đầu và các biến thể có cùng độ "mạnh" (power) - hiểu theo nghĩa, chúng đoán nhận được cùng lớp ngôn ngữ. Trong phần này, chúng ta tìm hiểu một vài trong số các biến thể và chứng minh sự tương đương kể trên. Chúng ta gọi tính bất biến trên là **tính vững** (robustness).

Để minh họa tính vững của mô hình máy Turing, ta sẽ thử thay đổi một chút trong định nghĩa của hàm chuyển. Trong định nghĩa của chúng ta, đầu đọc buộc phải di chuyển sang trái hoặc phải sau mỗi bước. Bây giờ, ta cho phép đầu đọc có thể đứng yên. Hàm chuyển khi đó sẽ có dạng

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$ trong đó: 'S' đại diện cho trạng thái đứng yên. Ta sẽ đặt ra câu hỏi liệu sự thay đổi này có thể làm cho máy Turing trở nên "mạnh mẽ hơn"? Câu trả lời là không. Ta có thể chuyển một máy Turing có thêm thuộc tính "đứng yên" về một máy Turing thông thường bằng cách thay mỗi bước đứng yên bởi 2 bước chuyển: chuyển phải rồi sau đó chuyển trái.

Ví dụ nhỏ trên minh họa ý tưởng chìa khóa để chứng minh sự tương đương giữa hai biến thể của máy Turing. Để chứng minh hai biến thể là tương đương, ta đơn giản chứng minh chúng chuyển được về nhau.

3.2.1 Biến thể 1: Máy Turing nhiều dải băng

Một **máy Turing nhiều dải băng** (multitape Turing machine) chỉ khác máy Turing thông thường ở chỗ có nhiều dải băng. Ứng mỗi dải băng, chúng ta lại có một đầu đọc/ghi. Ban đầu, đầu vào xuất hiện trên dải băng số 1, và những dải băng khác chỉ chứa kí tự trống. Hàm chuyển

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k$$

trong đó k là số lượng dải băng. Biểu thức

$$\delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, L, R, \ldots, L)$$

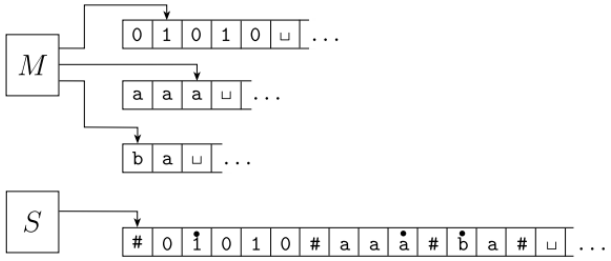
nghĩa là nếu máy ở trạng thái q_i và các đầu đọc 1 tới k đang đọc các kí tự a_1 tới a_k, tương ứng, máy sẽ chuyển sang trạng thái q_j, viết các kí tự b_1 đến b_k, và di chuyển mỗi đầu đọc tới các vị trí trái hoặc phải , tương ứng. Máy Turing nhiều dải băng có vẻ "mạnh mẽ" hơn máy Turing thông thường

(cũng như cảm giác ban đầu của ta rằng NFA có vẻ "mạnh mẽ" hơn DFA), nhưng chúng ta sẽ chỉ ra chúng tương đương. Nhắc lại rằng hai máy được gọi tương đương nếu chúng đoán nhận cùng ngôn ngữ.

Kết quả 3.2.1. *Mỗi máy Turing nhiều dải băng có một máy Turing một dải băng tương đương.*

Chứng minh. Như trong ví dụ, ta sẽ chỉ ra cách chuyển một máy Turing nhiều dải băng M về một máy Turing một dải băng S. (Chiều ngược lại là hiển nhiên)
Giả thiết M có k dải băng. Như vậy, ta mong muốn S phải lưu thông tin của k dải băng trên dải băng duy nhất của nó. Ta sẽ sử dụng một ký tự mới # làm kí tự phân chia để chia nội dung giữa các dải băng khác nhau. Bên cạnh nội dung của các dải băng này, S cũng phải "biết" vị trí của các đầu đọc trên từng dải băng của M. Nó cần kí tự đặc biệt để đánh dấu các vị trí đó, chẳng hạn, "viết" một dấu chấm lên đầu kí tự (ví dụ, a thành $a\dot{}$). Ta nghĩ về những việc trên như là thêm các đầu đọc và dải băng "ảo" cho S. Việc thêm các các dải băng và đầu đọc "ảo" thực chất là việc thêm các kí tự (#, các kí tự dải có thêm dấu '.' trên đầu) vào bảng chữ Γ của M để tạo thành Γ', bao gồm các ký tự được viết trên băng của S. Ta minh họa bằng hình 3.5.

Thuật toán cho S được mô tả như sau:
S : "Với xâu đầu vào $w = w_1 w_2 \dots w_n$ của M:



Hình 3.5: Minh họa việc chuyển máy Turing 3 dải băng về máy Turing thông thường. Nguồn [1]

1. Đầu tiên, S đưa vào dải băng của nó định dạng thể hiện k dải băng của M như phân tích trên.

$$\#w_1^{\cdot}w_2 \dots w_n\# \sqcup^{\cdot} \# \sqcup^{\cdot} \# \dots \#$$

2. S quét qua dải băng của nó từ dấu # đầu tiên(ở tận cùng bên trái) đến dấu # thứ $k+1$(tận cùng bên phải), để xác định những ký tự nào nằm dưới các đầu đọc ảo. Sau đó, S di chuyển để thay đổi dải băng tương ứng cách mà hàm chuyển của M quy định.

3. Nếu tại thời điểm nào đó, S di chuyển đầu đọc ảo nào đó về bên phải đến vị trí của một dấu #, điều đó có nghĩa M đã di chuyển đầu đọc tương ứng đến vị trí kí tự trống chưa được đọc trên dải băng tương ứng. Do đó, S tịnh tiến nội dung trên dải băng của nó từ vị trí đầu đọc ảo đang xét đến dấu # cuối cùng sang phải một ô, đồng thời viết một kí tự trống vào ô đang xét (ô lúc trước chứa dấu #)."

Một hệ quả trực tiếp của định lý trên là: □

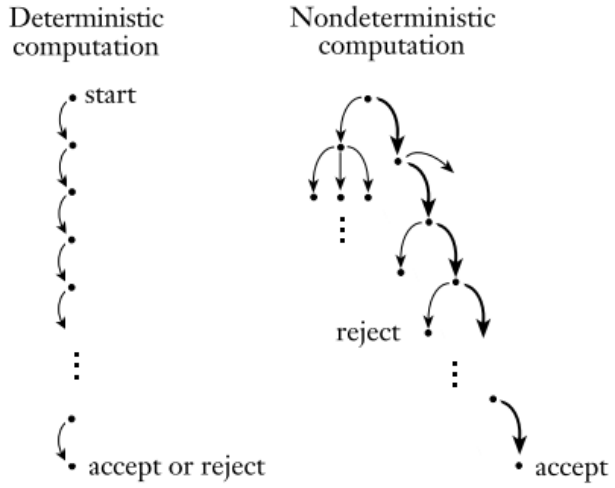
Một ngôn ngữ là Turing - đoán nhận được nếu và chỉ nếu có một máy Turing nhiều dải băng nào đó đoán nhận nó

3.2.2 Biến thể 2: Máy Turing đa định

Hàm chuyển của một máy Turing đa định (NTM) là hàm đa trị:

$$\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R\}}.$$

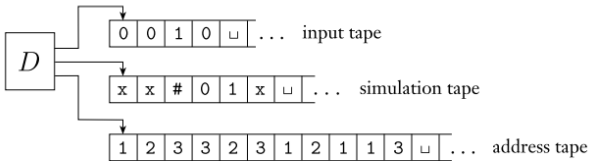
Quá trình tính toán của máy Turing đa định (nondeterministic Turing machine) có thể xem như một cây mà các nhánh của nó tương ứng với các khả năng (possibility) khác nhau (Hình 3.6). Nếu nhánh nào đó của sự tính toán này dẫn đến trạng thái chấp nhận, máy chấp nhận xâu đầu vào. Tập tất cả các xâu được chấp nhận bởi máy đa định M tạo thành ngôn ngữ của M, kí hiệu là $L(M)$.



Hình 3.6: Các quá trình tính toán đơn định và đa định với một nhánh chấp nhận. Nguồn [1]

Kết quả 3.2.2. *Nếu N là một máy Turing đa định, thì có một máy Turing đơn định D sao cho $L(N) = L(D)$.*

Chứng minh. Trước khi bắt tay vào xây dựng D, ta sẽ định nghĩa địa chỉ cho một nút trên cây tính toán đa định của N như sau. Đặt $b := \max_{(q,X)\in Q\times\Gamma} |\delta(q, X)|$. Mỗi nút trên cây của N có thể có nhiều nhất b con. Mỗi nút trên cây, ta gán địa chỉ là một xâu trên bảng chữ cái $T_b = \{1, 2, , \ldots, b\}$. Ví dụ, ta gán địa chỉ (2,3,1) cho nút mà ta đi đến bằng cách: bắt đầu từ gốc, đi tới con thứ 2 ν của gốc, đi tới con thứ 3 χ của ν, sau cùng đi tới con thứ nhất của χ. Như vậy, mỗi nút trên cây tương ứng với một xâu như trên, nhưng ngược lại, có những xâu trên T_b không ứng với nút nào trong trường hợp có những nút có ít hơn b con. Xâu rỗng sẽ là địa chỉ của gốc. Ta thiết kế D gồm ba dải băng. Theo định lý 3.2.1, ta đã biết máy Turing nhiều dải băng tương đương với máy Turing một dải băng. Máy D dùng ba dải băng của nó như minh họa trong hình 3.7.



Hình 3.7: Máy Turing đơn định D tương đương với máy đa định N. Nguồn [1]

1. Đầu tiên, dải 1 chứa xâu đầu vào w,và dải 2 và 3 đều rỗng.

2. Sao chép dải 1 vào dải 2 và khởi tạo dải 3 là xâu rỗng.

3. Ta xử lý xâu trên dải 2 bằng cách: thực hiện phép chuyển dựa trên cây tính toán đa định của N theo địa chỉ được lưu trên dải 3. Nếu địa chỉ không hợp lệ hoặc đã thực hiện xong phép chuyển và không nhận được một cấu hình chấp nhận, chuyển sang bước 4. Nếu nhận được một cấu hình, chấp nhận đầu vào và kết thúc.

4. Thay thế xâu trên dải 3 bởi xâu tiếp theo theo thứ tự trên xâu. Quay lại bước 2.

   Trong đó: thứ tự trên xâu là thứ tự, ưu tiên xâu ngắn hơn, và trong trường hợp hai xâu cùng độ dài thì nó là thứ tự từ điển.

□

Chương 4

Một số bài toán về máy Turing

Phần này sẽ giải quyết một số bài toán liên quan đến máy Turing.

4.1 Bài toán 1: Chuyển máy Turing nhiều dải băng về máy Turing một dải băng

Cho máy Turing k dải băng $M = (Q_M, \Sigma_M, \Gamma_M, \delta_M, q_{0M}, q_{aM}, q_{rM})$ với $\delta_M : Q \times \Gamma^k \mapsto Q \times \Gamma^k \times \{L, R\}^k$. Giả thiết $Q_M = \{q_1, \ldots, q_m\}, \Gamma_M = \{a_1, \ldots, a_n\}$. Thiết kế máy Turing 1 dải băng $D = (Q_D, \Sigma_D, \Gamma_D, \delta_D, q_{0D}, q_{aD}, q_{rD})$ tương đương với máy ban đầu như sau:

- $Q_D = Q_M \cup \{q_{i,a_{i_1},\ldots,a_{i_k}} | a_{i_j} \in \Gamma_M\}$
- $\Sigma_D = \Sigma_M \cup \Sigma_{M*}$ trong đó, Σ_{M*} là tập tất cả các kí tự có dạng $s^\bullet$ với

$s \in \Sigma_M$

- $\Gamma_D = \Gamma_M \cup \{\#\} \cup \Gamma_{M*}$ trong đó Γ_{M*} là tập tất cả các kí tự có dạng $s^\bullet$ với $s \in \Gamma_M$
- $q_{0D} = q_{0M}$
- $q_{aD} = q_{aM}$
- $q_{rD} = q_{rM}$

Hàm chuyển δ_D như sau:
Với $\delta_M(q_i, a_{i_1}, \ldots, a_{i_k}, D_1, \ldots, D_k) = (q_j, b_{i_1}, \ldots, b_{i_k})$ trong đó $D_t \in \{L, R\}$, ta có: $\delta_D(q_i, a^\bullet_{i_1}) = (q_{i,a_{i_1},\sqcup,\ldots,\sqcup}, b^\bullet_{i_1}, D_1)$,
$\delta_D(q_{i,a_{i_1},\ldots,a_{i_t},\sqcup,\ldots,\sqcup}, a^\bullet_{i_{t+1}}) = (q_{i,a_{i_1},\ldots,a_{i_{t+1}},\sqcup,\ldots,\sqcup}, b^\bullet_{i_t}, D_t)$
Không trong các trường hợp này, hàm chuyển chỉ đơn giản chuyển đầu đọc sang phải.

4.2 Bài toán 2: Chuyển máy Turing đa định về máy Turing đơn định

Cho máy Turing đa định $M = (Q_M, \Sigma_M, \Gamma_M, \delta_M, q_{0M}, q_{aM}, q_{rM})$ với $\delta_M : Q \times \Gamma_M \mapsto 2^{Q \times \Gamma_M \times \{L,R\}}$.
Giả thiết $Q_M = \{q_1, \ldots, q_m\}, \Gamma_M = \{a_1, \ldots, a_n\}$ Thiết kế máy Turing đơn định $D = (Q_D, \Sigma_D, \Gamma_D, \delta_D, q_{0D}, q_{aD}, q_{rD})$ tương đương với máy ban đầu như sau:

- $\Sigma_D = \Sigma_M \cup \Sigma_{M*}$ trong đó, Σ_{M*} là tập tất cả các kí tự có dạng $s^\bullet$ với $s \in \Sigma_M$
- $\Gamma_D = \Gamma_M$
- $q_{0D} = q_{0M}$
- $q_{aD} = q_{aM}$
- $q_{rD} = q_{rM}$

Đầu tiên, sao chép xâu từ dải 1 sang dải 2.
Sau đó, đọc từng kí tự trong dải 3 và chuyển đầu đọc của dải 2 theo kí tự đã đọc. Hàm chuyển δ_D như sau:
Với $\delta_M(q, a) = \{(q_1, a_1, D_1), \ldots, (q_k, a_k, D_k)\}$ trong đó $D_t \in \{L, R\}$, ta có: $\delta_D(q, s, a_{i_2}, i_3) = (q_{i_3}, s, a_{i_3}, i_3 + 1, S, D_{i_3}, R)$.
Không trong các trường hợp này, hàm chuyển chỉ đơn giản chuyển đầu đọc sang phải.

Chương 5

Ứng dụng máy Turing

Nói nôm na, một thuật toán (algorithm) là một tập các chỉ dẫn đơn giản để thực hiện một nhiệm vụ nào đó. Thuật toán đóng một vai trò quan trọng trong toán học. Từ thời cổ đại, người ta đã đưa ra các thuật toán(ví dụ, thuật chia Ơclit, sàng Ơ-ra-tot-xten). Tuy vậy, đến tận thế kỉ 20, khái niệm rõ ràng về thuật toán vẫn chưa được định nghĩa. Trước đó, các nhà toán học chỉ có một định nghĩa trực giác về thuật toán và sử dụng khái niệm đó khi mô tả các thuật toán. Nhưng định nghĩa trực giác trên không đủ để nhận được những hiểu biết sâu sắc về thuật toán. Ví dụ, bài toán số 10 của Hilbert đưa ra năm 1900 là có tồn tại hay không một thuật toán xác định liệu một đa thức có nghiệm nguyên. Ngày nay, ta biết rằng không tồn tại một thuật toán như thế. Tuy nhiên, với các nhà toán học thế kỉ 20, không thể chứng minh một thuật toán là không tồn tại mà không có một định nghĩa chính xác.

Nhờ lý thuyết máy Turing, Church và Turing đưa ra định nghĩa về thuật

toán cần thiết để giải bài toán số 10 của Hilbert. Năm 1970, Yuri Matijasevic đã chứng minh không tồn tại một thuật toán để kiểm tra liệu một đa thức có nghiệm nguyên.

Định đề Church-Turing:

Định nghĩa trực giác về thuật toán tương đương với thuật toán trên máy Turing.

Ví dụ bài toán số 10 của Hilbert phát biểu theo ngôn ngữ máy Turing: đặt

$$D = \{p | p \text{ là một đa thức có nghiệm nguyên.}\}$$

Bài toán số 10 của Hilbert hỏi rằng liệu D là Turing-quyết định được.

5.1 Máy Turing tính hàm

Máy Turing có thể được xem như một máy tính có khả năng tìm được giá trị của các hàm bộ phận. Để thấy điều đó, ta giả sử rằng máy Turing T khi đã cho xâu đầu vào x sẽ dừng lại với xâu y ở trên băng của nó. Khi đó ta có thể định nghĩa $T(x) = y$. Miền xác định của T là tập các xâu làm cho T sẽ dừng lại. $T(x)$ sẽ không xác định nếu T không dừng lại khi x đã cho như một đầu vào. Việc xem một máy Turing như một máy tính được cho các giá trị của một hàm trên các xâu là rất hữa ích.

Để xem một máy Turing như một máy tính các hàm từ tệp các bộ k số nguyên không âm đến tập các số nguyên không âm chúng ta cần phải có

một cách để biểu diễn các bộ k số nguyên trên băng của máy. Muốn vậy ta dùng biểu diễn nhị phân của các số nguyên (gồm các số 0 và 1).

Ví dụ: Xây dựng một máy Turing để tính tổng hai số nguyên:

$$f(x, y) = x + y$$

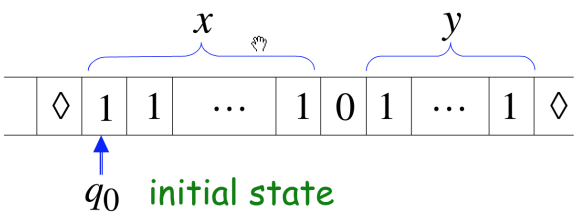
Với

Input String: $x0y$ unary

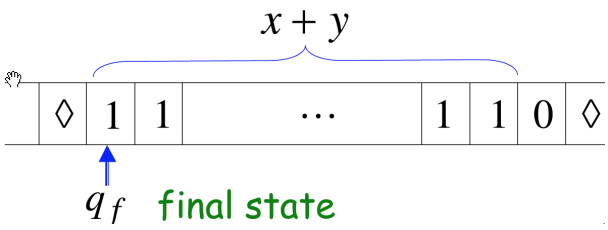
Output Srtring: $xy0$ unary

Máy Turing sẽ thực hiện với:

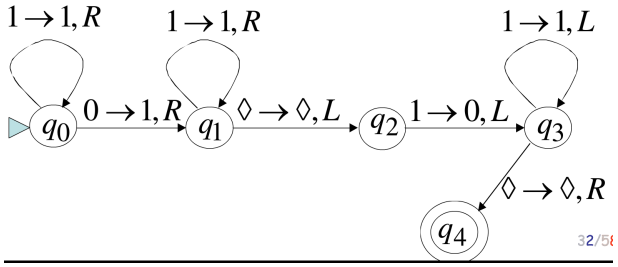
Trạng thái bắt đầu:



Trạng thái kết thúc:



Sơ đồ hàm chuyển như sau:



5.2 Tính liên thông của đồ thị

Với A là một ngôn ngữ gồm tất cả các chuỗi biểu thị các đồ thị vô hướng liên thông. Một đồ thị là liên thông nếu mọi nút của nó có thể đến được từ nút khác bằng cách di chuyển theo các cạnh của đồ thị.

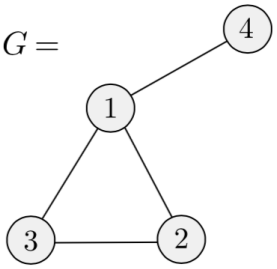
$$A = \{\langle G \rangle \mid G \text{ là đồ thị vô hướng liên thông}\}$$

Sau đây là miêu tả cho máy Turing M quyết định A

M = "Với đầu vào G, mã hoá cho đồ thị G:

1. Chọn nút đầu tiên của G và đánh dấu nó.
2. Lặp lại bước sau cho đến khi không có nút mới nào được đánh dấu:
3. Đối với mỗi nút trong G, đánh dấu nó nếu nó được gắn bởi một cạnh với một nút đã được đánh dấu.
4. Quét tất cả các nút của G để xác định xem tất cả chúng có được đánh dấu hay không. Nếu có, chấp nhận; nếu không từ chối."

Ví dụ: Một mã hoá của đồ thị G thành string với danh sách các nút và theo sau là danh sách các cạnh của G.

$G =$ 

$\langle G \rangle =$

```
(1,2,3,4)((1,2),(2,3),(3,1),(1,4))
```

Khi M nhận được đầu vào G, trước tiên, nó sẽ kiểm tra xem liệu đầu vào có phải là mã hóa phù hợp hay không. Để làm như vậy, M quét băng để chắc chắn rằng có hai danh sách và chúng ở dạng thích hợp. Danh sách đầu tiên phải là danh sách các số thập phân riêng biệt và danh sách thứ hai phải là danh sách các cặp số thập phân.
Sau đó M kiểm tra:
Đầu tiên, danh sách nút không được lặp lại
Thứ hai, mọi nút xuất hiện trên danh sách cạnh cũng sẽ xuất hiện trên danh sách nút. Nếu đầu vào vượt qua các kiểm tra này, thì đó là mã hóa của đồ thị G. Việc xác minh này hoàn thành kiểm tra đầu vào và M chuyển sang bước 1.

Đối với bước 1, M đánh dấu nút đầu tiên bằng dấu chấm.

Đối với giai đoạn 2, M quét danh sách các nút để tìm một nút không dấu và đánh dấu nó bằng cách gạch chân biểu tượng đầu tiên. Sau đó M quét lại danh sách để tìm một nút có chấm và gạch chân nó.

Bây giờ M quét danh sách các cạnh. Đối với mỗi cạnh, M kiểm tra xem hai nút được gạch chân n_1 và n_2 có phải là các nút xuất hiện trong cạnh đó hay không. Nếu đúng, M chấm n_1, xóa phần gạch chân và tiếp tục từ đầu bước 2. Nếu không, M sẽ kiểm tra cạnh tiếp theo trong danh sách. Nếu không còn cạnh nào nữa, $\{n_1, n_2\}$ không phải là cạnh của G. Sau đó M di chuyển phần gạch chân trên n_2 sang nút chấm tiếp theo và bây giờ gọi nút này là n_2. Nó lặp lại các bước trong đoạn này để kiểm tra, như trước đây, liệu cặp mới $\{n_1, n_2\}$ có phải là một cạnh hay không. Nếu không có thêm các nút chấm, n_1 không được gắn vào bất kỳ nút chấm nào. Sau đó M đặt các gạch chân sao cho n_1 là nút không dấu tiếp theo và n_2 là nút chấm đầu tiên và lặp lại các bước trong đoạn này. Nếu không còn các nút được phát hiện nữa, M đã không thể tìm thấy bất kỳ nút mới nào để chấm, vì vậy nó chuyển sang bước 4.

Đối với bước 4, M quét danh sách các nút để xác định xem tất cả có được chấm hay không. Nếu có, nó đi vào trạng thái chấp nhận.Nếu không, nó đi vào trạng thái từ chối. Điều này hoàn thành mô tả của TM M.

Chương 6

Kết luận

Báo cáo bắt đầu bằng việc giới thiệu cơ sở lý thuyết của ngôn ngữ, xâu và otomat hữu hạn. Trên cơ sở đó, ta tìm hiểu lý thuyết về máy Turing đơn định, đa định và các bài toán đưa các máy Turing phức tạp về đơn giản (máy Turing đa định, máy Turing nhiều dải băng về máy Turing đơn định) ngoài ra còn mở rộng thêm về các ứng dụng của máy Turing trên lý thuyết như giải quyết bài toán: Có tồn tại hay không nghiệm nguyên của một đa thức bất kỳ,....

Mặc dù nhóm đã cố gắng, nhưng do kiến thức của các thành viên còn hạn chế nên báo cáo vẫn còn nhiều thiếu sót, nhiều vấn đề chưa được giải quyết triệt để. Nhóm xin chân thành cảm ơn cô đã đưa ra nhiều góp ý quý báu để chúng em có thể hoàn thiện báo cáo này.

Tài liệu tham khảo

[1] Michael Sipser, *Introduction to the Theory of Computation*, 3rd Ed., Cengage Learning, 2012.

[2] John E.Hopcroft, Rajeev Motwani & Jeffrey D.Ullman, *Introduction to Automata Theory, Languages and Computation*, 2nd Ed, Addison-Wesley, 2001.