Tạp chí Tin học và Điều khiển học, T. 17, S. 1 (2001), 72–77

CÁC THUẬT TOÁN TÌM DẠNG CHUẨN CỦA VẾT VÀ VẾT ĐỒNG BỘ

HOÀNG CHÍ THÀNH

Abstract. The theory of traces, originated by A. Mazurkiewicz in 1977, is an attempt to provide a mathematical description of the behaviour of concurrent systems. The normal form of a trace gives an optimal concurrent order to perform the process described by the trace.
After composing two concurrent systems, the synchronization of corresponding traces languages describes the behaviour of the composed system whilst the synchronization of corresponding traces describes its processes. The paper investigates the synchronization of traces and points out algorithms for finding the normal form of trace and synchronization trace.
The results not only show an optimal performance of processes of the composed systems but also aid in constructing whole behaviours of these systems.

Tóm tắt. Ngôn ngữ vết là một trong các mô hình tốt để mô tả dáng điệu của các hệ tương tranh, trong đó mỗi vết biểu diễn một quá trình. Dạng chuẩn của vết cho phương án tối ưu để thực hiện các hành động trong quá trình một cách tương tranh. Nội dung bài báo là xây dựng thuật toán đệ qui ngắn gọn để tìm dạng chuẩn của vết, nghiên cứu sự đồng bộ của các vết, dạng biểu diễn và dạng chuẩn của vết đồng bộ.

1. MỞ ĐẦU

Vết và ngôn ngữ do A. Mazurkiewicz đề xuất là một công cụ tốt để mô hình hóa các quá trình và dáng điệu của các hệ tương tranh. Mỗi một vết đều có duy nhất một dạng chuẩn, mà dạng chuẩn này chỉ ra cách thực hiện tối ưu cho quá trình được mô tả bởi vết. Ngoài những thuật toán đã có để tìm dạng chuẩn của vết [1], chúng tôi xây dựng một thuật toán đệ qui rất ngắn gọn, giúp tìm nhanh dạng chuẩn của vết. Khái niệm đồng bộ vết là một khái niệm mang ý nghĩa thực tế rất quan trọng, đặc biệt khi xây dựng hệ lớn hợp thành trực tiếp từ dáng điệu của các hệ thành phần [3]. Bài báo cũng tập trung nghiên cứu các thuật toán nhanh tìm dạng chuẩn của vết đồng bộ, góp phần nghiên cứu dáng điệu và các tính chất của hệ hợp thành.

2. VẾT VÀ DẠNG CHUẨN CỦA VẾT

Giả sử A là một bảng chữ cái hữu hạn.

Định nghĩa 2.1.

1) Một quan hệ đối xứng và không phản xạ $p \subseteq A \times A$ được gọi là một *quan hệ độc lập* trên A.

2) Một *bảng chữ cái tương tranh* là một cặp (A, p) trong đó A là một bảng chữ cái và p là một quan hệ độc lập trên A.

Ký hiệu: $Æ = (A, p)$.

Bảng chữ cái tương tranh thường được biểu diễn bởi môt đồ thị vô hướng.

Giả sử $Æ = (A, p)$ một bảng chữ cái tương tranh.

Quan hệ $=_p \subseteq A^* \times A^*$ được định nghĩa như sau:

$$\forall x, y \in A^* : \quad x =_p y \Leftrightarrow \exists x_1, x_2 \in A^*, \ (a, b) \in p : x = x_1\, ab\, x_2 \wedge y = x_1\, ba\, x_2$$

Hai từ được gọi là bằng nhau theo p nếu từ này được tạo bởi từ kia bằng cách giao hoán hai chữ cái độc lập đứng liền nhau. Nếu mỗi chữ cái thể hiện một hành động nào đó, mỗi từ thể hiện một dãy các hành động thì thứ tự thực hiện các hành động độc lập đứng liền nhau là không quan

trọng. Do vậy chúng có thể được thực hiện một cách tương tranh với nhau.

Từ quan sát này chúng ta đưa ra một quan hệ tương đương như sau:

Quan hệ $\equiv_p \subseteq A^* \times A^*$ được định nghĩa là quan hệ tương đương nhỏ nhất trên A^* chứa $=_p$.

Với mọi $x, y \in A^*$, x và y được gọi là p - tương đương nếu và chỉ nếu $x \equiv_p y$.

Ta có:

1) $\equiv_p = (=_p)^*$.

2) $x \equiv_p y \Rightarrow |x| = |y|$.

Hai từ là p - tương đương sẽ có cùng độ dài vì các chữ cái trong chúng là như nhau.

Định nghĩa 2.2. Giả sử Æ $= (A, p)$ là một bảng chữ cái tương tranh.

1) Mỗi một lớp tương đương của quan hệ tương đương $\equiv_p$ được gọi là một *vết* trên Æ. Độ dài của vết t được xác định bởi độ dài của từ đại diện của nó. Nếu $t = [x]_p$ thì: $|t| = |x|$.

2) Một tập các vết được gọi là một *ngôn ngữ vết* trên Æ.

Như vậy, mỗi vết bao gồm tất cả các cách thực hiện tuần tự có thể của một dãy các hành động được biểu diễn bởi đại diện của vết này.

Các phép toán trên vết:

1. Nếu t_1, t_2 là các vết, $t_1 = [x_1]_p$ và $t_2 = [x_2]_p$ thì hợp thành của t_1 và t_2 là:

$$t_1 \,.\, t_2 = [\, x_1 \,.\, x_2 \,]_p \,.$$

2. Nếu T_1, T_2 là các ngôn ngữ vết thì hợp thành của T_1 và T_2 là:

$$T_1 \,.\, T_2 = \{t_1 \,.\, t_2 \,|\, t_1 \in T_1 \wedge T_2 \in T_2\}.$$

2.1. Dạng chuẩn của vết

Nhờ phép toán hợp thành vết, mỗi một vết có nhiều cách phân rã thành hợp thành của một số vết khác. Chúng ta quan tâm tới một phân rã đặc biệt như sau:

Với mỗi vết $t = [w]$ phân rã: $t = t_1 \,.\, t_2 \,. \ldots . t_m$, $m \geq 0$ mà:

1) t_i không rỗng.

2) $t_i = [u_i]$, mỗi chữ cái trong u_i chỉ xuất hiện một lần và hai chữ cái trong u_i là độc lập với nhau (giao hoán được).

3) Nếu $t_i = [u_i]$, $t_{i+1} = [u_{i+1}]$ thì mỗi chữ cái trong u_{i+1} không độc lập với một chữ cái nào đó trong u_i (độc lập cực đại).

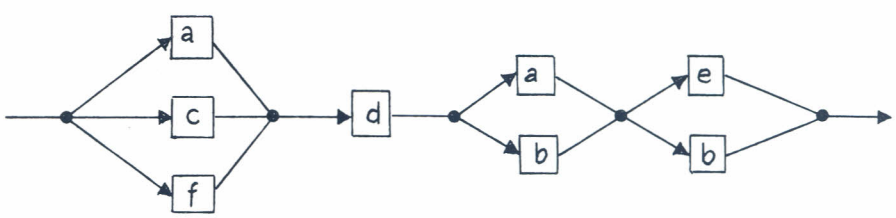
Phân rã này được gọi là *dạng chuẩn* của vết t.

2.2. Ý nghĩa của dạng chuẩn của vết

Giả sử vết t trên một bảng chữ cái tương tranh Æ có dạng chuẩn là:

$$t = [\, a\, c\, f \,] \,.\, [\, d \,] \,.\, [\, a\, b \,] \,.\, [\, e\, b \,]_p \,.$$

Sự thực hiện của quá trình được mô tả bởi vết t có thể biểu diễn bởi sơ đồ sau đây:



Dạng chuẩn của vết cho phương án thực hiện một cách tương tranh tối ưu của quá trình được biểu diễn bởi vết này.

Định lý 2.3. *Với mọi vết t, dạng chuẩn của nó đều tồn tại và duy nhất.*

2.3. Các thuật toán tìm dạng chuẩn của vết

Thuật toán 2.4. [1]

Thuật toán sẽ đọc từ đại diện của vết t từ trái sang phải. Dùng một mảng các con trỏ r mà nếu a_j xuất hiện thì nó sẽ được ghép vào từ $u(r(j))$. Mảng các con trỏ được cập nhật để đảm bảo rằng mỗi chữ cái không độc lập với a_j nếu xuất hiện thì chỉ có thể đặt vào trong các khối sau đó.

1. Với mọi $1 \leq i \leq n$, $r(i) := 1$ (n là số các chữ cái trong bảng chữ cái A).
2. Với mọi $1 \leq k \leq l$, $u(k) := \lambda$ (l là độ dài của vết t).
3. $k := 0$.
4. $k := k + 1$.
5. Đặt j mà $u(k) = a_j$.
6. $u(r(j)) := u(r(j)).a_j$.
7. Với mọi $1 \leq i \leq n$ mà $i \neq j$, $r(i) \leq r(j)$ và $(a_i, a_j) \notin p$, $r(i) := r(j) + 1$.
8. $r(j) := r(j) + 1$.
9. Nếu $k < l$ thì chuyển lên bước 4.
10. Chọn m mà $u(m) \neq \lambda$ và
    hoặc là $m < l$ nhưng $u(m + 1) = \lambda$ hoặc $m = l$.

Thuật toán 2.5.

Trước hết chúng ta nhắc lại khái niệm sau đây:

Đồ thị phụ thuộc của vết $t = [w]$ là đồ thị phụ thuộc của từ w, ký hiệu bởi $D(w)$, được định nghĩa đệ qui như sau:

- $D(\lambda)$ là đồ thị rỗng.
- $D(w.a)$ là đồ thị được tạo bởi $D(w)$ thêm một đỉnh mới a và các cạnh hướng tới đỉnh mới này từ các đỉnh của $D(w)$ mà chúng phụ thuộc vào a.

Với vết t chúng ta xây dựng đồ thị phụ thuộc của nó.

Từ đồ thị phụ thuộc lấy ra tập các đỉnh cực tiểu bao gồm các đỉnh không có cạnh nào đi tới, đó chính là thành phần u_1.

Loại bỏ tập các đỉnh cực tiểu cùng các cạnh đi ra từ chúng, ta được một đồ thị phụ thuộc mới, lại lấy tập các đỉnh cực tiểu của đồ thị này ta được thành phần u_2.

Lặp lại các bước trên cho đến khi nhận được một đồ thị rỗng, chúng ta sẽ nhận được tất cả các thành phần trong dạng chuẩn của vết.

Từ chính định nghĩa của dạng chuẩn của vết, chúng ta đưa ra một thuật toán đệ qui sau đây.

Thuật toán 2.6. (Đệ qui)

Giả sử $t = [w]$ và $[u_1].[u_2] \ldots [u_m]$ là dạng chuẩn cần tìm của vết t.

Khi đó $t = [u_1].[u_2] \ldots [u_m] = [u_1\, u_2 \ldots u_m] = [u_1].[u_2 \ldots u_m]$.

Như vậy các chữ cái trong u_1 được "dịch chuyển" từ phía phải về đầu từ đại diện w, hay nói cách khác: các chữ cái trong u_1 phải "giao hoán được" với tất cả các chữ cái bên trái nó.

Tìm được u_1, ta có:

$t = [u_1].[w']$, với w' là từ nhận được từ w sau khi loại bỏ các chữ cái trong u_1 đã dịch chuyển được về phía trái.

Sau đó ta lặp lại bước làm ở trên để tìm u_2 gồm các chữ cái trong w' mà chúng "giao hoán được" với tất cả các chữ cái bên trái.

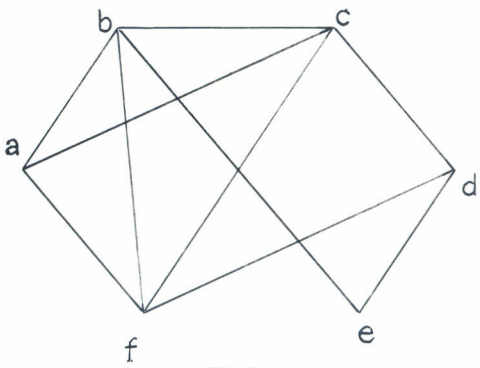
Lặp lại các bước trên cho đến khi nhận được từ đại diện rỗng, ta sẽ nhận được tất cả các thành phần trong dạng chuẩn của t.

So với hai thuật toán ở trên, Thuật toán 2.6 có cùng độ phức tạp như Thuật toán 2.5 là $O(l^2)$ nhưng thuật toán này đơn giản hơn rất nhiều.

Ví dụ. Cho bảng chữ cái tương tranh Æ được biểu diễn bằng đồ thi vô hướng (hình 1) và vết $t = [adcfaebb]_p$.

Theo Thuật toán 2.4 chúng ta xây dưng mảng các con trỏ r như sau:

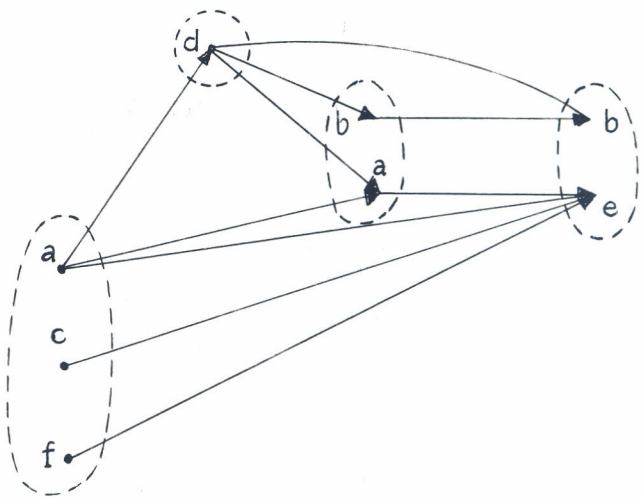
w	$r(1)$	$r(2)$	$r(3)$	$r(4)$	$r(5)$	$r(6)$
$adcfaebb$	$\underline{1}$	1	1	1	1	1
$dcfaebb$	2	1	1	$\underline{2}$	2	1
$cfaebb$	3	3	$\underline{1}$	3	2	1
$faebb$	3	3	2	3	2	$\underline{1}$
$aebb$	$\underline{3}$	3	2	3	2	2
ebb	4	3	2	4	$\underline{4}$	2
bb	5	$\underline{3}$	5	4	5	5
b	5	$\underline{4}$	5	4	5	5



Hình 1

Từ đó ta tìm được dạng chuẩn của vết $t = [acf].[d].[ab].[eb]$.

Theo Thuật toán 2.5, đồ thị phụ thuộc của vết t và các tập đỉnh cực tiểu sau mỗi bước loại bỏ như sau:



Từ các tập đỉnh cực tiểu, chúng ta tìm được dạng chuẩn của vết t giống như kết quả của Thuật toán 2.4.

Theo Thuật toán 2.6, các chữ cái c và f có thể chuyển sang trái, ta được thành phần đầu tiên $u_1 = acf$. Không có phần tử nào có thể chuyền sang trái để kết hợp với d, vậy $u_2 = d$. Bây giờ b có thể chuyển sang trái để kết hợp với a, nên $u_3 = ab$, còn lại $u_4 = eb$.

$$adcfaebb \rightarrow acf|d|aebb \rightarrow acf|d|ab|eb$$

Dạng chuẩn của vết được ứng dụng nhiều trong các bài toán thiết kế, điều khiển... chẳng hạn trong việc điều khiển các cơ sở dữ liệu phân tán, trong dây chuyền sản xuất.

3. ĐỒNG BỘ CỦA CÁC QUÁ TRÌNH

Giả sử Æ = (A, p) là một bảng chữ cái tương tranh. Quan hệ $d = A \times A \setminus p$ là một quan hệ phản xạ, đối xứng và được gọi là quan hệ *phụ thuộc* trên Æ.

Việc sử dụng quan hệ độc lập p hay quan hệ phụ thuộc d theo đúng nghĩa của chúng trong hầu hết các trường hợp là như nhau.

Mỗi vết trên Æ biểu diễn một *quá trình* và một ngôn ngữ vết trên Æ biểu diễn một *hoạt động*.

Giả sử

$$Æ_1 = (A_1, p_1) \text{ và } Æ_2 = (A_2, p_2) \text{ là hai bảng chữ cái tương tranh.}$$

Vậy $d_1 = A_1 \times A_1 \setminus p_1$ và $d_2 = A_2 \times A_2 \setminus p_2$ là hai quan hệ phụ thuộc tương ứng.

3.1. Phép chiếu vết

Định nghĩa 3.1. Giả sử t là một vết trên Æ. Khi đó hình chiếu của t trên d_2 được định nghĩa như sau:

$$[\lambda]_{d_1|d_2} = [\lambda]_{d_1 \cap d_2}$$

$$(t.[a]_{d_1})_{|d_2} = \begin{cases} (t_{|d_2}).[a]_{d_1 \cap d_2} & \text{nếu } a \in A_2 \\ t_{|d_2} & \text{nếu } a \in A_1 \setminus A_2 \end{cases}$$

3.2. Phép hợp thành tương tranh của các quá trình

Bảng chữ cái tương tranh $Æ = (A_1 \cup A_2, p)$, với $p = p_1 \cup p_2$ là bảng chữ cái tương tranh hợp thành từ $Æ_1$ và $Æ_2$.

Định nghĩa 3.2. Giả sử t_1 là một vết trên $Æ_1$ và t_2 là một vết trên $Æ_2$. Phép *hợp thành tương tranh* của t_1 và t_2, kí hiệu $t_1 \# t_2$ là vết t trên Æ mà

$$t_{|d_1} = t_1 \text{ và } t_{|d_2} = t_2 .$$

Nếu vết t tồn tại thì:

1) Mỗi một hành động xuất hiện trong t phải là một hành động của t_1 hoặc t_2 và không có hành động nào khác xuất hiện trong t.

2) Sự xuất hiện của các hành động sắp xếp trong t_1 và t_2 cũng được sắp xếp giống hệt như trong t.

3) Số lần xuất hiện của một hành động nào đó trong t_1 hoặc trong t_2 cũng giống hệt như trong t.

Do đó t được gọi là *đồng bộ* của t_1 và t_2 .

Định lý 3.3. *Đồng bộ của t_1 và t_2 tồn tại khi và chỉ khi $t_{1|d_2} = t_{2|d_1}$.*

4. DẠNG CHUẨN CỦA ĐỒNG BỘ

Để tìm dạng chuẩn của đồng bộ trước hết phải tìm từ đại diện cho vết đồng bộ, sau đó áp dụng các Thuật toán 2.4 - 2.6 đã trình bày trong phần 2 để tìm dạng chuẩn cho vết đồng bộ này.

4.1. Phép tổ hợp song song của các ngôn ngữ

Giả sử A là một bảng chữ cái và $B \subseteq A$.

$h_B : A^* \to B^*$ là *đồng cấu xóa* được định nghĩa như sau:

$$\forall a \in A, \quad h_B(a) = \begin{cases} a & \text{nếu } a \in B, \\ \lambda & \text{nếu } a \notin B. \end{cases}$$

Nếu $x = a_1 a_2 .... a_k \in A^*$ thì:

$$h_B(x) = h_B(a_1) h_B(a_2) ... h_B(a_k)$$

và thường được viết là: $x_{|B}$ - hình chiếu của x trên B.

Với mỗi ngôn ngữ L, kí hiệu $\vec{L}$ là bảng chữ cái nhỏ nhất xây dựng lên L.

Định nghĩa 4.1. Giả sử L_1 và L_2 là hai ngôn ngữ. *Tổ hợp song song* của hai ngôn ngữ L_1, L_2 là một ngôn ngữ, kí hiệu là $L_1 \| L_2$ được định nghĩa như sau:

$$L_1 \| L_2 = \{x \mid x \in (\overline{L}_1 \cup \overline{L}_2)^* \wedge x_{|\overline{L}_1} \in L_1 \wedge x_{|\overline{L}_2} \in L_2\}.$$

Định lý 4.2. *Giả sử* $t_1 = [w_1]_{d_1}$ *và* $t_2 = [w_2]_{d_2}$, *nếu đồng bộ* t *của* t_1 *và* t_2 *tồn tại thì* $w = w_1 \| w_2$ *là từ đại diện của* t, *nghĩa là:* $t = [w]_{d_1 \cup d_2}$.

Giả sử $t_1 = [u]_{d_1}$, $t_2 = [v]_{d_2}$ và tồn tại đồng bộ t của t_1 và t_2. Khi đó: $t_{1|d_2} = t_{2|d_1} = [a_1 a_2 ... a_k]_{d_1 \cap d_2}$. với $a_1, a_2, ..., a_k \in A_1 \cap A_2$.

Có thể khai triển

$u = u_0 a_1 u_1 a_2 u_2 ... u_{k-1} a_k u_k$ với $u_i \in (A_1 \setminus A_2)^*$,

$v = v_0 a_1 v_1 a_2 v_2 ... v_{k-1} a_k v_k$ với $v_i \in (A_2 \setminus A_1)^*$.

Từ Định lý 3.3 và Định lý 4.2 chúng ta có:

Hệ quả 4.3. *Nếu* t *là đồng bộ của* t_1 *và* t_2 *thì:*

$w = u_0 v_0 a_1 u_1 v_1 a_2 ... u_{k-1} v_{k-1} a_k u_k v_k$ *là một từ đại diện của* t.

Hệ quả trên cho chúng ta một phương pháp hữu hiệu để tìm từ đại diện của vết đồng bộ, để từ đó tìm nhanh được dạng chuẩn của vết đồng bộ. Kết quả này còn được dùng để xác định dáng điệu của các hệ hợp thành từ chính dáng điệu của các hệ thành phần.

TÀI LIỆU THAM KHẢO

[1] J. I. Albersberg and G. Rozenberg, Theory of Traces, *Theoretical Computer Science* **60** (1988) 1–82.

[2] A. Mazurkiewicz, *Concurrent Program Schemes and their Interpretation*, DAIMI Report PB-78, Aarhus University, Denmark, 1977.

[3] Hoang Chi Thanh, *Behavioural Synchronization of Net Systems*, IMSc Report 115, Madras, India, 1991, 136–145.

[4] P. S. Thiagarajan, Some behavioural aspects of net theory, *Theoretical Computer Science* **71** (1990) 133–153.

Nhận bài ngày 3 - 7 - 2000

Trường Đại học Khoa học tự nhiên
Đại học Quốc gia Hà Nội