

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

*

LUẬN VĂN THẠC SỸ
KHOA CÔNG NGHỆ THÔNG TIN

Học viên thực hiện : **Lê Văn Hùng**

Giảng viên hướng dẫn : **PGS. TS. Phạm Ngọc Hùng**

Chương 1

Giới thiệu

Chương 2

Kiến thức cơ sở

2.1 Dạng đặc tả sử dụng hệ chuyển trạng thái được gắn nhãn

Định nghĩa 2.1: Hệ chuyển trạng thái được gắn nhãn (Labelled Transition System - LTS [2])

Một LTS là một bộ có thứ tự gồm 4 thành phần: $M = \langle Q, \Sigma, \delta, q_0 \rangle$

Trong đó:

- $Q = \{q_0, q_1, \dots, q_n\}$ là tập các trạng thái,
- $\Sigma = \{w_0, w_1, \dots, w_n\}$ là tập các sự kiện,
- $\delta \subseteq Q \times \Sigma \times Q$ là hàm chuyển trạng thái, và
- $q_0 \in Q$ là trạng thái khởi tạo.

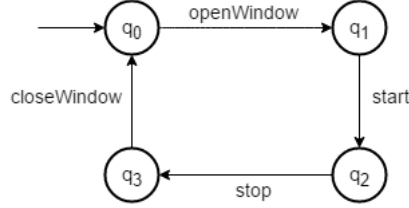
Ta kí hiệu $q_i \xrightarrow{w_i} q_j$ nếu và chỉ nếu có một sự kiện w_i chuyển hệ thống từ trạng thái q_i sang trạng thái q_j , khi đó $(q_i, w_i, q_j) \in \delta$. Điều này có nghĩa khi một hệ thống đang ở trạng thái q_i , nếu có một sự kiện w_i xảy ra thì hệ thống sẽ chuyển sang trạng thái q_j . Tương tự, khi hệ thống đang ở trạng thái q_j nếu có một hành động w_k xảy ra thì hệ thống sẽ chuyển sang trạng thái q_k . Như vậy, chuỗi hai hành động $q_i \xrightarrow{w_i} q_j, q_j \xrightarrow{w_k} q_k$ có thể chuyển hệ thống từ trạng thái q_i sang trạng thái q_k . Khi đó, ta có thể kí hiệu $q_i \xrightarrow{w_i w_k} q_k$.

Ví dụ 2.1: Ví dụ về một hệ thống chuyển trạng thái được gắn nhãn.

Trên hình 2.1 là một ví dụ về một LTS $M = \langle Q, \Sigma, \delta, q_0 \rangle$, trong đó:

- $Q = \{q_0, q_1, q_2, q_3\}$,
- $\Sigma = \{openWindow, start, stop, closeWindow\}$,

- $\delta = \{(q_0, openWindow, q_1), (q_1, start, q_2), (q_2, stop, q_3), (q_3, closeWindow, q_0)\}$,
và
- q_0 là trạng thái khởi tạo.



Hình 2.1: Một hệ thống chuyển trạng thái được gán nhãn.

Định nghĩa 2.2: Kích thước của một tập hợp [1].

Kích thước của một tập hợp $Q = \{q_0, q_1, \dots, q_n\}$ là số phần tử của tập hợp Q , kí hiệu là $|Q|$.

Ví dụ 2.2: Với LTS được cho bởi hình 2.1, tập các trạng thái $Q = \{q_0, q_1, q_2, q_3\}$ nên $|Q| = 4$.

Định nghĩa 2.3: Kích thước của một LTS [3].

Kích thước của một LTS $M = \langle Q, \Sigma, \delta, q_0 \rangle$ là số trạng thái của M , kí hiệu là $|M|$, trong đó $|M| = |Q|$.

Ví dụ 2.3: Với LTS được cho bởi hình 2.1, kích thước của LTS đó là $|M| = |Q| = 4$.

Định nghĩa 2.4: Vết của LTS.

Vết của một LTS $M = \langle Q, \Sigma, \delta, q_0 \rangle$ là một chuỗi hữu hạn các sự kiện có dạng $\sigma = w_0 w_1 \dots w_k$ với $w_k \in \Sigma$ và $0 \leq k \leq n$ sao cho tồn tại trạng thái $q_i \in Q$ mà $q_0 \xrightarrow{\sigma} q_i$.

Như vậy, vết σ của LTS M là một chuỗi các sự kiện có thể quan sát được mà M có thể thực hiện được từ trạng thái khởi tạo q_0 .

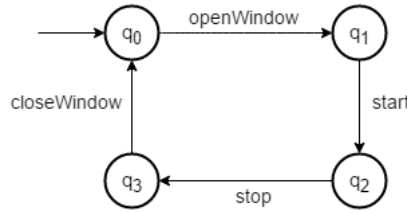
Ví dụ 2.4: Vết của LTS.

Hình 2.2 minh họa một LTS $M = \langle Q, \Sigma, \delta, q_0 \rangle$, trong đó:

- $Q = \{q_0, q_1, q_2, q_3\}$,
- $\Sigma = \{openWindow, start, stop, closeWindow\}$,
- $\delta = \{(q_0, openWindow, q_1), (q_1, start, q_2), (q_2, stop, q_3), (q_3, closeWindow, q_0)\}$,
và
- q_0 là trạng thái khởi tạo.

Ta thấy, chuỗi các hành động $openWindow\ start\ stop$ là một vết của M , bởi vì tại trạng thái khởi tạo q_0 , khi sự kiện $openWindow$ xảy ra, hệ thống chuyển

sang trạng thái q_1 , tiếp tục xảy ra sự kiện *start* hệ thống chuyển sang trạng thái q_2 , khi xảy ra sự kiện *stop* hệ thống chuyển sang trạng thái q_3 . Chuỗi các hành động *openWindow start stop* chuyển hệ thống từ trạng thái khởi tạo q_0 sang trạng thái $q_3 \in Q$ nên chuỗi các hành động *openWindow start stop* là một vết của LTS. Tương tự, chuỗi các hành động *openWindow, openWindow start, openWindow start stop closeWindow, openWindow start stop closeWindow openWindow, ...* đều là vết của M.



Hình 2.2: Minh họa vết của LTS.

Định nghĩa 2.5: Ngôn ngữ của LTS.

Ngôn ngữ của LTS M kí hiệu là $L(M)$ được định nghĩa như sau:

$$L(M) = \{\alpha \mid \alpha \text{ là một vết của } M\}$$

Ví dụ 2.5: Ví dụ về ngôn ngữ của LTS.

Với LTS M như ở hình 2.2, ngôn ngữ của M là:

$$L(M) = \{openWindow, openWindow start, openWindow start stop, \dots\}$$

2.2 Dạng đặc tả sử dụng hàm logic (Bool)

Định nghĩa 2.6: Hàm logic [1].

$B = \{T, F\}$ là miền giá trị logic. Với X là tập hợp các biến logic, một hàm logic $\theta(X)$ được định nghĩa $\theta(X): B^{|X|} \rightarrow B$.

Ví dụ 2.6: Ví dụ về hàm logic.

Với X là tập hợp gồm 3 phần tử, $X = \{x, y, z\}$ trong đó $x, y, z \in B$. Hàm logic $\theta(x, y, z) = x \wedge y \vee z$ chính là một ánh xạ $\theta(X): B^3 \rightarrow B$.

Định nghĩa 2.7: Phép gán [1].

Với X là tập hợp các biến logic, phép gán v được định nghĩa $v: X \rightarrow B$.

Ví dụ 2.7: Với X là tập hợp gồm 3 phần tử, $X = \{x, y, z\}$ trong đó $x, y, z \in B$, $v_1(x) = T$, $v_2(x) = F$, $v_1(y) = T$, $v_2(y) = F$, $v_1(z) = T$ và $v_2(z) = F$, ... là các phép gán trên tập X.

Định nghĩa 2.8: Phép gán hàm [1].

Với $\phi(X)$ là hàm một logic trên tập X, v là một phép gán trên tập X, phép gán

hàm kí hiệu $\phi[v]$ là kết quả thu được khi thay các phần tử $x \in X$ bởi $v(x)$. Với X và X' là các tập biến logic, trong đó $X' = \{x' \mid x \in X\}$, $\psi(X, X')$ là hàm logic trên hai tập X và X' , với $v(x)$ và $v'(x')$ lần lượt là các phép gán trên tập X và X' , kí hiệu $\psi[v, v']$ là kết quả thu được khi thay một cách tương ứng các phần tử $x \in X$ bởi $v(x)$ và $x' \in X'$ bởi $v'(x')$.

Ví dụ 2.8: Với $X = \{x\}$, $X' = \{x'\}$ là tập hợp các biến logic, $\phi(x) = \bar{x}$ là một hàm logic trên tập X . Nếu $v(x) = T$ thì $\phi[v] = F$ và nếu $v(x) = F$ thì $\phi[v] = T$. Với $\psi(x, x') = x \vee x'$ là một hàm logic trên tập X và X' , nếu $v(x) = T$, $v'(x') = F$ thì $\psi[v, v'] = T \vee F = T$.

Một cách tổng quát, với n tập các biến logic X, X_1, X_2, \dots, X_n trong đó $X_i = \{x_i \mid x \in X\}$, $\psi(X, X_1, X_2, \dots, X_n)$ là hàm logic tương ứng trên các tập biến logic X, X_1, X_2, \dots, X_n , ta kí hiệu $\psi[v_1, v_2, \dots, v_n]$ là kết quả thu được khi thay một cách tương ứng các phần tử $x_1 \in X_1$ bởi $v_1(x_1)$, $x_2 \in X_2$ bởi $v_2(x_2)$, ... và $x_n \in X_n$ bởi $v_n(x_n)$.

Định nghĩa 2.9: Dạng đặc tả sử dụng hàm logic.

Dạng đặc tả sử dụng hàm logic là một bộ có thứ tự gồm 4 phần tử:

$$N = \langle X, E, \tau(X, E, X'), \iota(X) \rangle$$

Trong đó:

- X là tập các biến logic dùng để biểu diễn các trạng thái của hệ thống. $X = \{x_0, x_1, \dots, x_n\}$,
- E là tập các biến logic dùng để biểu diễn các hành vi của hệ thống. $E = \{e_0, e_1, \dots, e_n\}$,
- $\tau(X, E, X')$ là hàm logic biểu diễn việc chuyển trạng thái của hệ thống, và
- $\iota(X)$ là hàm logic dùng để biểu diễn các trạng thái khởi tạo của hệ thống.

Ví dụ 2.9: Ví dụ về dạng đặc tả sử dụng hàm logic.

Với dạng đặc tả sử dụng hàm logic $N = \langle X, E, \tau(X, E, X'), \iota(X) \rangle$.

Trong đó:

- $X = \{x_1, x_2\}$, $X' = \{x_5, x_6\}$,
- $E = \{x_3, x_4\}$,
- $\tau(X, E, X') = (\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge \bar{x}_4 \wedge x_5 \wedge \bar{x}_6) \mid (x_1 \wedge \bar{x}_2 \wedge x_3 \wedge \bar{x}_4 \wedge \bar{x}_5 \wedge x_6) \mid (\bar{x}_1 \wedge x_2 \wedge \bar{x}_3 \wedge x_4 \wedge \bar{x}_5 \wedge \bar{x}_6)$, và
- $\iota(X) = \bar{x}_1 \wedge \bar{x}_2$.

Định nghĩa 2.10: Vết của dạng đặc tả sử dụng hàm logic.

Với dạng đặc tả sử dụng hàm logic $N = \langle X, E, \tau(X, E, X'), \iota(X) \rangle$, v là phép gán

cho hàm biểu diễn trạng thái trên tập X , γ là phép gán cho hàm biểu diễn sự kiện trên tập E , một chuỗi hữu hạn $\xi = \gamma^0 \gamma^1 \dots \gamma^n$ được gọi là vết của N khi và chỉ khi tồn tại tập các phép gán $v^0, v^1, \dots, v^{n+1}, \gamma^0, \gamma^1, \dots, \gamma^n$ sao cho $\iota[v^0] = T$ và $\tau[v^i, \gamma^i, v^{i+1}] = T$ với $0 \leq i \leq n$.

Ví dụ 2.10: Ví dụ về vết của dạng đặc tả sử dụng hàm logic.

Cho dạng đặc tả sử dụng hàm logic $N = \langle X, E, \tau(X, E, X'), \iota(X) \rangle$.

Trong đó:

- $X = \{x_1, x_2\}, X' = \{x_5, x_6\},$
- $E = \{x_3, x_4\},$
- $\tau(X, E, X') = \{(\bar{x}_1 \wedge \bar{x}_2) \wedge (\bar{x}_3 \wedge \bar{x}_4) \wedge (x_5 \wedge \bar{x}_6) \mid (x_1 \wedge \bar{x}_2) \wedge (x_3 \wedge \bar{x}_4) \wedge (\bar{x}_5 \wedge x_6) \mid (\bar{x}_1 \wedge x_2) \wedge (\bar{x}_3 \wedge x_4) \wedge (\bar{x}_5 \wedge \bar{x}_6)\},$ và
- $\iota(X) = \bar{x}_1 \wedge \bar{x}_2.$

Vì $\iota(X) = \bar{x}_1 \wedge \bar{x}_2$, với v^0 là phép gán trên tập X sao cho $v^0(x_1) = F$ và $v^0(x_2) = F$ nên $\iota[v^0] = T \wedge T = T$. Mặt khác, gọi v^1 là phép gán trên tập X sao cho $v^1(x_1) = T, v^1(x_2) = F, \gamma^0$ là phép gán trên tập E sao cho $\gamma^0(x_3) = F$ và $\gamma^0(x_4) = F$. Khi đó, $\tau[v^0, \gamma^0, v^1] = T$ nên $\xi = \gamma^0$ là một vết của N . Mặt khác, với phép gán v^1, γ^1, v^2 sao cho $v^1(x_1) = T, v^1(x_2) = F, \gamma^1(x_3) = T, \gamma^1(x_4) = F, v^2(x_5) = F, v^2(x_6) = T$ thì $\tau[v^1, \gamma^1, v^2] = T$. Do đó $\xi = \gamma^0 \gamma^1$ cũng là một vết của N . Một cách hoàn toàn tương tự chúng ta có thể tìm được các vết tiếp theo của N .

Định nghĩa 2.11: Ngôn ngữ của dạng đặc tả sử dụng hàm logic.

Cho dạng đặc tả sử dụng hàm logic $N = \langle X, E, \tau(X, E, X'), \iota(X) \rangle$, tập hợp tất cả các vết của N được gọi là ngôn ngữ của N và được kí hiệu là $L(N)$. Ta có: $L(N) = \{ \xi \mid \xi \text{ là một vết của } N \}.$

Ví dụ 2.11: Với dạng đặc tả sử dụng hàm logic N cho bởi ví dụ 2.10 thì ngôn ngữ của N là $L(N) = \{FF, FFTF, FFTFFT, \dots\}$

2.3 Bảng ánh xạ

Bảng ánh xạ (mapping) là một bảng dùng để lưu lại các ánh xạ khi chuyển đổi từ dạng đặc tả sử dụng LTS sang dạng đặc tả sử dụng hàm logic và ngược lại. Gọi Map là kí hiệu của bảng ánh xạ. Với một LTS $M = \langle Q, \Sigma, \delta, q_0 \rangle$ trong đó $Q = Q_1 \cup Q_2$ với Q_1 là tập các trạng thái đầu vào, Q_2 là tập các trạng thái đầu ra và một dạng đặc tả sử dụng hàm logic $N = \langle X, E, \tau(X, E, X'), \iota(X) \rangle$, trong đó $X = X_1 \cup X_2$ với X_1 là tập các biến logic dùng để biểu diễn các trạng thái đầu vào của hệ thống, X_2 là tập các biến logic biểu diễn các trạng thái đầu ra

của hệ thống. Ta định nghĩa:

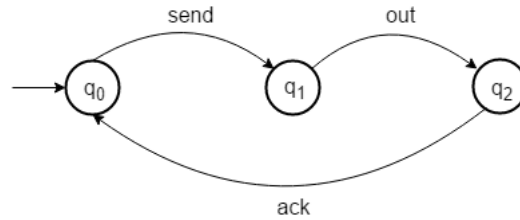
$$Map = \begin{cases} Q_1 \mapsto X_1 \\ Q_2 \mapsto X_2 \\ \Sigma \mapsto E \\ \delta(q, w, q') \mapsto \tau(v, \gamma, v') \end{cases}$$

Trong đó:

- $Q_1 \mapsto X_1$ là một song ánh từ tập các trạng thái đầu vào đến tập các biến logic dùng để biểu diễn các trạng thái đầu vào của hệ thống,
- $Q_2 \mapsto X_2$ là một song ánh từ tập các trạng thái đầu ra đến tập các biến logic dùng để biểu diễn các trạng thái đầu ra của hệ thống,
- $\Sigma \mapsto E$ là một song ánh từ tập các sự kiện đến tập các biến logic dùng để biểu diễn các sự kiện, và
- $\delta(q, w, q') \mapsto \tau(v, \gamma, v')$ là một ánh xạ từ hàm chuyển trạng thái đến tập các hàm logic dùng để biểu diễn việc chuyển trạng thái của hệ thống.

Ví dụ 2.12: Ví dụ về bảng ánh xạ.

Cho LTS $M = \langle Q, \Sigma, \delta, q_0 \rangle$ như trên hình 2.3, trong đó:



Hình 2.3: Ví dụ về một LTS.

- $Q = \{q_0, q_1, q_2\}$, $Q_1 = \{q_0, q_1, q_2\}$, $Q_2 = \{q_1, q_2, q_0\}$,
- $\Sigma = \{send, out, ack\}$,
- $\delta = \{(q_0, send, q_1), (q_1, out, q_2), (q_2, ack, q_0)\}$, và
- q_0 là trạng thái khởi tạo.

Và dạng đặc tả sử dụng hàm logic $N = \langle X, E, \tau(X, E, X'), \iota(X) \rangle$.

Trong đó:

- $X = \{x_1, x_2, x_5, x_6\}$,

- $E = \{x_3, x_4\}$,
- $\tau(X, E, X') = \{(\bar{x}_1 \wedge \bar{x}_2) \wedge (\bar{x}_3 \wedge \bar{x}_4) \wedge (x_5 \wedge \bar{x}_6) \mid (x_1 \wedge \bar{x}_2) \wedge (x_3 \wedge \bar{x}_4) \wedge (\bar{x}_5 \wedge x_6) \mid (\bar{x}_1 \wedge x_2) \wedge (\bar{x}_3 \wedge x_4) \wedge (\bar{x}_5 \wedge \bar{x}_6)\}$, và
- $\iota(X) = \bar{x}_1 \wedge \bar{x}_2$.

Khi đó, nếu dạng đặc tả sử dụng LTS M và dạng đặc tả dụng hàm logic N là tương đương. Ta có bảng ánh xạ:

Q_1	q_0	q_1	q_2
X_1	$\bar{x}_1 \wedge \bar{x}_2$	$x_1 \wedge \bar{x}_2$	$\bar{x}_1 \wedge x_2$

Bảng 2.1: Thành phần $Q_1 \mapsto X_1$ trong bảng ánh xạ

Q_2	q_1	q_2	q_0
X_2	$x_5 \wedge \bar{x}_6$	$\bar{x}_5 \wedge x_6$	$\bar{x}_5 \wedge \bar{x}_6$

Bảng 2.2: Thành phần $Q_2 \mapsto X_2$ trong bảng ánh xạ

Σ	send	out	ack
E	$\bar{x}_3 \wedge \bar{x}_4$	$x_3 \wedge \bar{x}_4$	$\bar{x}_3 \wedge x_4$

Bảng 2.3: Thành phần $\Sigma \mapsto E$ trong bảng ánh xạ

$\delta(q, e, q')$	$\delta(a, send, b)$	$\delta(b, out, c)$	$\delta(c, ack, a)$
$\tau(v, \gamma, v')$	$x_0 x_1 \wedge x_2 x_3 \wedge x_4 \bar{x}_5$	$x_0 \bar{x}_1 \wedge x_2 \bar{x}_3 \wedge x_4 \bar{x}_5$	$\bar{x}_0 x_1 \wedge \bar{x}_2 x_3 \wedge x_4 x_5$

Bảng 2.4: Thành phần $\delta(q, e, q') \mapsto \tau(v, \gamma, v')$ trong bảng ánh xạ

Chương 3

Phương pháp chuyển đổi

3.1 Thuật toán

Đầu vào (Input): Một LTS.

Đầu ra (Output): Dạng đặc tả sử dụng hàm logic và bảng ánh xạ.

Thuật toán chia làm 4 bước:

- Bước 1: Mã hóa tập các trạng thái đầu vào - Thuật toán 3.1.
- Bước 2: Mã hóa tập các trạng thái đầu ra - Thuật toán 3.1.
- Bước 3: Mã hóa tập các sự kiện - Thuật toán 3.1.
- Bước 4: Mã hóa tập các chuyển trạng thái - Thuật toán 3.2.

3.1.1 Thuật toán mã hóa một tập hợp

Tập các trạng thái đầu vào, tập các trạng thái đầu ra hay tập các sự kiện gọi chung là một tập hợp. Vì các bước tiến hành mã hóa một tập hợp là giống nhau nên luận văn chỉ trình bày một thuật toán chung. Khi tiến hành mã hóa thì tùy từng mục đích mã hóa tập đầu vào sẽ thay đổi. Cụ thể, nếu chúng ta tiến hành mã hóa tập các trạng thái đầu vào Q_1 thì đầu vào cho thuật toán mã hóa sẽ là tập các trạng thái đầu vào Q_1 , hoặc nếu chúng ta tiến hành mã hóa tập các trạng thái đầu ra Q_2 thì đầu vào cho thuật toán mã hóa sẽ là tập các trạng thái đầu ra Q_2 và nếu chúng ta tiến hành mã hóa tập các sự kiện Σ thì đầu vào cho thuật toán mã hóa sẽ là tập các sự kiện Σ .

Thuật toán 3.1: Thuật toán mã hóa một tập hợp

Đầu vào: Một tập hợp A (A là tập các trạng thái đầu vào hoặc A là tập các trạng thái đầu ra hoặc A là tập các sự kiện)

Đầu ra : Tập hợp các phần tử của tập hợp A đã được mã hóa và bảng ánh xạ

```
1  if  $|A| = 1$  then
2    |    $z = 1$ 
3  else
4    |   if  $\log_2(|A|)$  là số nguyên then
5    |   |    $z = \log_2(|A|)$ 
6    |   else
7    |   |    $z = \lceil \log_2(|A|) \rceil + 1$ 
8    |   end
9  end
10 for mỗi phần tử  $a_i$  trong tập A do
11   |   Lưu  $a_i$  vào bảng ánh xạ ứng với thành phần A
12   |    $\alpha_i = \text{True}$ 
13   |    $k = \text{Thứ tự của } a_i \text{ trong A}$ 
14   |   Chuyển  $k$  sang số nhị phân với độ dài  $z$  bit
15   |   for mỗi bit trong chuỗi nhị phân biểu diễn  $k$  do
16   |   |   if  $bit = 0$  then
17   |   |   |    $\alpha_i = \alpha_i \wedge \bar{x}_j$ 
18   |   |   else
19   |   |   |    $\alpha_i = \alpha_i \wedge x_j$ 
20   |   |   end
21   |   end
22   |   Lưu  $\alpha_i$  vào bảng ánh xạ ứng với vị trí của phần tử  $a_i$  trong thành
      |   phần A
23 end
```

Với A là tập các phần tử cần mã hóa $A = \{a_0, a_1, \dots, a_n\}$. Từ bước (1) - (9), ta sẽ xác định được số biến logic cần dùng để mã hóa tất cả các phần tử của tập hợp A, gọi z là số biến logic cần dùng. Gọi $X = \{x_1, x_2, \dots, x_z\}$ là tập các biến logic dùng để mã hóa các phần tử của A. Mỗi phần tử a_i trong A sẽ được biểu diễn dưới dạng $x_1 \wedge x_2 \wedge \dots \wedge x_z$. Xét một phần tử bất kỳ a_i trong tập A, theo bước (13) ta xác định được thứ tự của a_i trong tập A là k , theo bước (14) thì k sẽ biểu diễn dưới dạng số nhị phân z bit. Theo bước (15) - (20) chúng ta sẽ mã hóa được phần tử a_i . Thêm vào đó, theo bước (22) dạng biểu diễn của phần tử a_i sẽ được lưu vào trong bảng ánh xạ. Bởi vì, thứ tự của mỗi phần tử

trong a_i là duy nhất nên số nhị phân biểu diễn thứ tự của phần tử a_i cũng sẽ là duy nhất, vì thế dạng mã của phần tử a_i là duy nhất. Một cách tương tự cho các phần tử khác trong A, sau khi vòng lặp ở bước (1) kết thúc chúng ta được dạng mã hóa được tất cả các trạng thái của tập A. Thêm vào đó, sau khi mã hóa, các thông tin về các trạng thái và dạng mã hóa tương ứng của từng trạng thái này đều được lưu vào bảng ánh xạ. Chúng ta sẽ thấy $\alpha_i = \alpha_i \bigwedge_{j=1}^n x_j$ là một hàm logic biểu diễn phần tử a_i và các phần tử x_j biểu diễn hàm α_i đều thuộc tập X.

Độ phức tạp: Độ phức tạp của thuật toán là $O(n)$, trong đó n là kích thước của tập A cần mã hóa.

3.1.2 Thuật toán mã hóa tập các chuyển trạng thái

Thuật toán 3.2: Mã hóa tập các chuyển trạng thái

Đầu vào: Tập hợp các chuyển trạng thái của LTS.

Đầu ra : Hàm chuyển trạng thái $\tau(X, E, X')$

```

1 for mỗi chuyển trạng thái được biểu diễn  $\delta(q, w, q')$  do
2     Lấy dạng biểu diễn  $\alpha_i$  của  $q$  từ bảng ánh xạ của tập các trạng thái
        đầu vào
3     Lấy dạng biểu diễn  $e_i$  của  $w_i$  từ bảng ánh xạ của tập các sự kiện
4     Lấy dạng biểu diễn  $\alpha_{i+1}$  của  $q'$  từ bảng ánh xạ của tập các trạng
        thái đầu ra
5      $\tau = \bigvee_{i=1}^n \{\alpha_i \wedge e_i \wedge \alpha_{i+1}\}$ 
6 end
7 return  $\tau$ 
```

Một chuyển trạng thái được biểu diễn là một bộ ba (q, w, q') , nên để mã hóa cho mỗi chuyển trạng thái, chúng ta cần tìm dạng mã hóa cho từng thành phần q , w và q' . Dựa theo thuật toán 3.1, chúng ta đã có được bảng ánh xạ lưu thông tin về các trạng thái và dạng mã hóa tương ứng, thông tin về các sự kiện và dạng mã hóa tương ứng. Xét một chuyển trạng thái (q_1, w_i, q_{i+1}) . Theo bước (2) q_i sẽ tương ứng với x_i trong bảng ánh xạ của tập các trạng thái đầu vào. Theo bước (3) sự kiện w_i sẽ tương ứng với e_i trong bảng ánh xạ của tập các sự kiện. Theo bước (4) trạng thái q_{i+1} sẽ tương ứng với x_{i+1} trong bảng ánh xạ của tập các trạng thái đầu ra. Sau bước (5), chuyển trạng thái (q_i, w_i, q_{i+1}) sẽ được biểu diễn dưới dạng $\alpha_i \wedge e_i \wedge \alpha_{i+1}$.

Độ phức tạp: Độ phức tạp của thuật toán là $O(n)$, trong đó n là kích thước của tập hợp các chuyển trạng thái cần mã hóa.

3.2 Chứng minh

Với thuật toán mã hóa đưa ra ở chương phía trước chúng ta hoàn toàn có thể chuyển đổi một dạng đặc tả sử dụng LTS sang dạng đặc tả sử dụng hàm logic và ngược lại từ dạng đặc tả sử dụng hàm logic kết hợp với bảng ánh xạ để chuyển đổi sang dạng đặc tả sử dụng LTS. Tuy nhiên, hai dạng đặc tả này sau khi chuyển đổi liệu có tương đương với nhau? Để chứng minh ngôn ngữ của dạng đặc tả được biểu diễn bằng LTS tương đương với ngôn ngữ của dạng đặc tả sử dụng hàm logic, chúng ta cần chứng minh 2 mệnh đề:

- Mệnh đề 1: Ngôn ngữ của dạng đặc tả hệ thống biểu diễn bằng LTS sau khi chuyển đổi sang dạng đặc tả biểu diễn bằng hàm logic được đoán nhận bởi dạng đặc tả sử dụng hàm logic. (1)
- Mệnh đề 2: Ngôn ngữ của dạng đặc tả sử dụng hàm logic sau khi chuyển sang dạng đặc tả biểu diễn bằng LTS được đoán nhận bởi LTS.

Chứng minh mệnh đề 1:

Với một LTS $M = \langle Q, \Sigma, \delta, q_0 \rangle$, $L(M)$ là ngôn ngữ của M và một dạng đặc tả sử dụng hàm logic $N = \langle X, E, \tau(X, E, X'), \iota(X) \rangle$, $L(N)$ là ngôn ngữ của N , N được chuyển đổi từ M . Với $\alpha = w_0 w_1 \dots w_n$ là vết của LTS M , áp dụng thuật toán 3.1 ta thu được $\xi = \gamma^0 \gamma^1 \dots \gamma^n$, ta cần chứng minh ξ được đoán nhận bởi $L(N)$. Thật vậy, xét q_0 là trạng thái khởi tạo của M , áp dụng thuật toán 3.1, chúng ta sẽ mã hóa q_0 thành α_0 . Vì α_0 là một hàm logic nên tồn tại một phép gán v^0 cho hàm α_0 trên tập X sao cho $\iota[v^0] = T$. (*)

Mặt khác, gọi $\delta(q_i, w_i, q_{i+1})$ là một chuyển trạng thái bất kì trong tập các chuyển trạng thái của LTS, áp dụng thuật toán 3.2, $\delta(q_i, w_i, q_{i+1})$ sẽ được mã hóa thành $\tau(\alpha_i, e_i, \alpha_{i+1})$. Vì $\alpha_i, e_i, \alpha_{i+1}$ là hội của các biến logic nên tồn tại các phép gán v^i cho hàm α_i trên tập X , v^{i+1} cho hàm α_{i+1} trên tập X' , γ^i là phép gán cho hàm e_i trên tập E sao cho $v^i = T$, $v^{i+1} = T$, $\gamma^i = T$ để $\tau[v^i, \gamma^i, v^{i+1}] = T$ (**). Từ (*), (**) và định nghĩa 2.10, $\xi = \gamma^0 \gamma^1 \dots \gamma^n$ là một vết của N hay $\xi \in L(N)$.

Chứng minh mệnh đề 2:

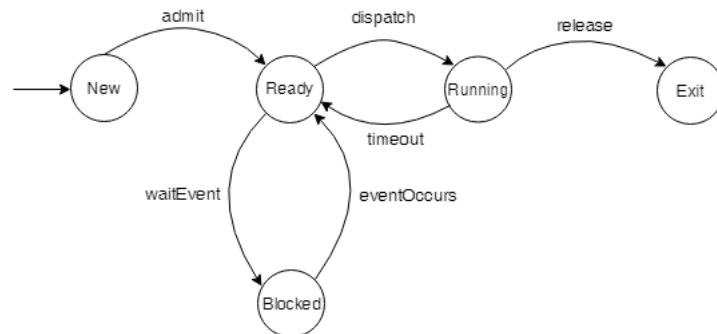
Với một LTS $M = \langle Q, \Sigma, \delta, q_0 \rangle$, $L(M)$ là ngôn ngữ của M và một dạng đặc tả sử dụng hàm logic $N = \langle X, E, \tau(X, E, X'), \iota(X) \rangle$, $L(N)$ là ngôn ngữ của N và N được chuyển đổi từ M . Gọi Map là bảng ánh xạ lưu các ánh xạ khi chuyển đổi từ M sang N . Gọi α_i là dạng mã hóa của trạng thái q_i . Gọi $\xi = \gamma^0 \gamma^1 \dots \gamma^n$ là một vết của N . Trong đó, γ^i là phép gán cho hàm e^i biểu diễn hành vi thứ i của hệ thống. Theo định nghĩa vết của dạng đặc tả đặc tả sử dụng hàm logic 2.10 vì $\xi = \gamma^0 \gamma^1 \dots \gamma^n$ là một vết của N nên tồn tại các phép gán v^0, v^1, \dots ,

$v^{n+1}, \gamma^0, \gamma^1, \dots, \gamma^n$ sao cho $\iota[v^0] = T$ và $\tau[v^i, \gamma^i, v^{i+1}] = T$ với $\forall i: 0 \leq i \leq n$.
 Vì $\iota[v^0] = T$ nên $\alpha_0[v^0] = T$ mà α_0 lại là dạng mã hóa của trạng thái q_0 nên q_0 là trạng thái khởi tạo của M. Thêm vào đó, vì $\tau[v^i, \gamma^i, v^{i+1}] = T$ nên dựa vào bảng ánh xạ Map ta thu được các hàm chuyển trạng thái $\delta(q_i, w_i, q_{i+1})$ tương ứng. Vì $0 \leq i \leq n$ nên ta có chuỗi các sự kiện $\sigma = w_0 w_1 \dots w_i$ là một vết của M. Với q_0 là trạng thái khởi tạo, sau khi sự kiện w_0 xảy ra hệ thống chuyển sang trạng thái q_1 (do sự tồn tại của chuyển trạng thái $\delta(q_0, w_0, q_1)$), từ trạng thái q_1 khi xảy ra sự kiện w_1 hệ thống chuyển sang trạng thái q_2 (do sự tồn tại của chuyển trạng thái $\sigma(q_1, w_1, q_2)$), một cách tương tự với các sự kiện từ w_2 đến w_{i-1} , sau khi sự kiện w_{i-1} xảy ra hệ thống chuyển sang trạng thái q_i , khi sự kiện w_i xảy ra, do sự tồn tại của chuyển trạng thái $\sigma(q_i, w_i, q_{i+1})$ nên hệ thống chuyển sang trạng thái q_{i+1} . Vì thế, với $\forall i: 0 \leq i \leq n$, tồn tại trạng thái $q_i \in Q$ sao cho $q_0 \xrightarrow{\sigma} q_i$ nên theo định nghĩa 2.4, $\sigma = w_0 w_1 \dots w_n$ là một vết của M.

Chương 4

Ví dụ về việc chuyển đổi giữa các dạng đặc tả

4.1 Giới thiệu về hệ thống



Hình 4.1: Một hệ thống chuyển trạng thái được gán nhãn.

Hình 4.1 là một LTS biểu diễn việc chuyển trạng thái của một tiến trình. Tiến trình là một chương trình đang được thực hiện. Một tiến trình có 5 trạng thái cơ bản:

- New là trạng thái khi tiến trình vừa được tạo hay nói cách khác là tiến trình vừa được sinh ra.
- Ready là trạng thái tiến trình được đưa vào hệ thống và được cấp phát đầy đủ tài nguyên (trừ CPU) để xử lý.
- Blocked là trạng thái mà tiến trình chờ đợi để được cấp phát thêm tài nguyên hoặc để một số sự kiện nào đó xảy ra, hay một quá trình vào/ra

kết thúc. Khi sự kiện đang chờ xảy ra thì tiến trình sẽ ở trạng thái Ready.

- Running là trạng thái mà tiến trình đang được sở hữu CPU để hoạt động, hay nói cách khác là các chỉ thị của tiến trình đang được thực hiện/ xử lý bởi processor.
- Exit là trạng thái mà tiến trình kết thúc việc xử lí.

Dưới đây là bảng mô tả việc chuyển đổi giữa các trạng thái trong tiến trình.

Trạng thái bắt đầu	Trạng thái kết thúc	Sự kiện	Nguyên nhân
-	New	-	Tiến trình đang được tạo
New	Ready	admit	Tiến trình được khởi tạo, đưa vào hệ thống và được cấp phát đầy đủ tài nguyên chỉ thiếu CPU
Ready	Running	dispatch	Tiến trình được cấp CPU để bắt đầu thực hiện/xử lý.
Running	Blocked	waitEvent	Tiến trình đang chờ một sự kiện nào đó xảy ra hay đang chờ một thao vào/ra kết thúc hay tài nguyên mà tiến trình yêu cầu chưa được hệ điều hành đáp ứng.
Blocked	Ready	eventOccurs	Sự kiện mà tiến trình chờ đã xảy ra, thao tác vào/ra mà tiến trình đợi đã kết thúc, hay tài nguyên mà tiến trình yêu cầu đã được hệ điều hành đáp ứng.
Running	Ready	timeout	Khi tiến trình đang chạy bị chiếm chỗ bởi tiến trình khác có độ ưu tiên cao hơn hoặc tiến trình đang chạy đã sử dụng quá thời gian cho phép
Running	Exit	Release	Tiến trình kết thúc

Bảng 4.1: Các chuyển trạng thái của tiến trình

4.2 Chuyển đổi dạng đặc tả sử dụng LTS sang dạng đặc tả sử dụng hàm logic

Xét LTS $M = \{Q, \Sigma, \delta, q_0\}$ như hình 4.1.

Trong đó:

- $Q = \{\text{New, Ready, Running, Exit, Blocked}\}$, $Q_1 = \{\text{New, Ready, Running, Blocked}\}$, $Q_2 = \{\text{Ready, Running, Exit, Blocked}\}$,
- $\Sigma = \{\text{admit, dispatch, timeout, release, waitEvent, eventOccurs}\}$,
- $\delta = \{(\text{New, admit, Ready}), (\text{Ready, dispatch, Running}), (\text{Running, release, Exit}), (\text{Running, timeout, Ready}), (\text{Ready, waitEvent, Blocked}), (\text{Blocked, eventOccurs, Ready})\}$, và
- New là trạng thái khởi tạo.

Chúng ta sẽ mã hóa tập các trạng thái và tập các sự kiện dựa theo thuật toán 3.1. Với đầu vào là tập các trạng thái đầu vào $Q_1 = \{\text{New, Ready, Running, Blocked}\}$, ta có $|Q_1| = 4$, vì $\log_2(4)$ là số nguyên nên theo bước (4) và bước (5) ta có $z = \log_2(4) = 2$ biến. Theo bước (10) chúng ta tiến hành mã hóa từng trạng thái trong tập Q_1 , trạng thái a_0 là trạng thái New, theo bước (11), chúng ta sẽ lưu New vào trong bảng ánh xạ ứng với thành phần $Q_1 \mapsto X_1$, theo bước (12), khởi tạo $\alpha_0 = \text{True}$, theo bước (13), k là vị trí của trạng thái New trong tập Q_1 nên $k = 0$, theo bước (14), k sẽ được biểu diễn dưới dạng số nhị phân có độ dài 2 bit (z bit) hay $k = 00$. Theo bước (15), xét bit thứ 1 của k , giá trị của bit này bằng 0 nên theo bước (17) $\alpha_0 = \alpha_0 \wedge \bar{x}_1$, bit tiếp theo là bit thứ 2, giá trị của bit này bằng 0, nên theo bước (17) $\alpha_0 = \alpha_0 \wedge \bar{x}_1 \wedge \bar{x}_2 = \bar{x}_1 \wedge \bar{x}_2$. Theo bước (22), chúng ta sẽ lưu α_0 vào trong bảng ánh xạ ứng với vị trí của trạng thái New trong thành phần $Q_1 \mapsto X_1$. Một cách hoàn toàn tương tự, chúng ta thu được dạng mã hóa của trạng thái Ready là $\alpha_1 = x_1 \wedge \bar{x}_2$, dạng mã hóa của trạng thái Running là $\alpha_2 = \bar{x}_1 \wedge x_2$ và dạng mã hóa của trạng thái Blocked là $\alpha_3 = x_1 \wedge x_2$. Sau khi mã hóa xong tập các trạng thái Q_1 ta thu được thành phần $Q_1 \mapsto X_1$ của bảng ánh xạ như bảng 4.1. Đồng thời sau khi mã hóa

Q_1	New	Ready	Running	Blocked
X_1	$\bar{x}_1 \wedge \bar{x}_2$	$x_1 \wedge \bar{x}_2$	$\bar{x}_1 \wedge x_2$	$x_1 \wedge x_2$

Bảng 4.2: Thành phần $Q_1 \mapsto X_1$ trong bảng ánh xạ

xong tập các trạng thái đầu vào Q_1 , chúng ta cũng thu được hàm logic dùng để biểu diễn trạng thái khởi tạo của hệ thống (cũng chính là dạng mã hóa của trạng thái q_0) là $\iota(X) = \bar{x}_1 \wedge \bar{x}_1$. Một cách hoàn toàn tương tự, áp dụng thuật toán 3.1 với đầu vào là tập các trạng thái đầu ra $Q_2 = \{\text{Ready, Running, Exit, Blocked}\}$ và tập các sự kiện $\Sigma = \{\text{admit, dispatch, timeout, release, waitEvent, eventOccurs}\}$ ta thu các thành phần $Q_2 \mapsto X_2$ và thành phần $\Sigma \mapsto E$ của bảng ánh xạ như sau:

Q_2	Ready	Running	Exit	Blocked
X_2	$\bar{x}_6 \wedge \bar{x}_7$	$x_6 \wedge \bar{x}_7$	$\bar{x}_6 \wedge x_7$	$x_6 \wedge x_7$

Bảng 4.3: Thành phần $Q_2 \mapsto X_2$ trong bảng ánh xạ

Σ	admit	dispatch	release
E	$\bar{x}_3 \wedge \bar{x}_4 \wedge \bar{x}_5$	$x_3 \wedge \bar{x}_4 \wedge \bar{x}_5$	$\bar{x}_3 \wedge x_4 \wedge \bar{x}_5$
Σ	timeout	waitEvent	eventOccurs
E	$x_3 \wedge x_4 \wedge \bar{x}_5$	$\bar{x}_3 \wedge \bar{x}_4 \wedge x_5$	$x_3 \wedge \bar{x}_4 \wedge x_5$

Bảng 4.4: Thành phần $\Sigma \mapsto E$ trong bảng ánh xạ

Công việc còn lại sẽ là mã hóa tập các chuyển trạng thái. Áp dụng thuật toán 3.2 với đầu vào là tập các chuyển trạng thái $\delta = \{(\text{New}, \text{admit}, \text{Ready}), (\text{Ready}, \text{dispatch}, \text{Running}), (\text{Running}, \text{release}, \text{Exit}), (\text{Running}, \text{timeout}, \text{Ready}), (\text{Ready}, \text{waitEvent}, \text{Blocked}), (\text{Blocked}, \text{eventOccurs}, \text{Ready})\}$. Theo bước (1) xét lần lượt từng chuyển trạng trong tập δ , với chuyển trạng thái đầu tiên (New, admit, Ready), theo bước (2), ta lấy được dạng biểu diễn của New từ thành phần $Q_1 \mapsto X_1$ của bảng ánh xạ là $\bar{x}_1 \wedge \bar{x}_2$. Theo bước (3), ta lấy được dạng biểu diễn của sự kiện admit từ thành phần $\Sigma \mapsto E$ của bảng ánh xạ là $\bar{x}_3 \wedge \bar{x}_4 \wedge \bar{x}_5$. Theo bước (4), ta lấy được dạng biểu diễn của Ready từ thành phần $Q_2 \mapsto X_2$ của bảng ánh xạ là $\bar{x}_6 \wedge \bar{x}_7$. Theo bước (5), ta thu được dạng mã hóa của chuyển trạng thái (New, admit, Ready) là $\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge \bar{x}_4 \wedge \bar{x}_5 \wedge \bar{x}_6 \wedge \bar{x}_7$. Áp dụng một cách hoàn toàn tương tự cho các chuyển trạng thái khác trong tập δ , cuối cùng ta thu được $\tau(X, E, X') = \{\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge \bar{x}_4 \wedge \bar{x}_5 \wedge \bar{x}_6 \wedge \bar{x}_7\} \mid \{x_1 \wedge \bar{x}_2 \wedge x_3 \wedge \bar{x}_4 \wedge \bar{x}_5 \wedge x_6 \wedge \bar{x}_7\} \mid \{\bar{x}_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge \bar{x}_5 \wedge \bar{x}_6 \wedge x_7\} \mid \{\bar{x}_1 \wedge x_2 \wedge \bar{x}_3 \wedge \bar{x}_4 \wedge x_5 \wedge \bar{x}_6 \wedge \bar{x}_7\} \mid \{x_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge x_4 \wedge \bar{x}_5 \wedge x_6 \wedge x_7\} \mid \{x_1 \wedge x_2 \wedge x_3 \wedge \bar{x}_4 \wedge x_5 \wedge \bar{x}_6 \wedge \bar{x}_7\}$. Và thành phần $\delta(q, e, q') \mapsto \tau(v, \gamma, v')$ của bảng ánh xạ:

$\delta(q, e, q')$	(New, admit, Ready)	(Ready, dispatch, Running)
$\tau(v, \gamma, v')$	$\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge \bar{x}_4 \wedge \bar{x}_5 \wedge \bar{x}_6 \wedge \bar{x}_7$	$x_1 \wedge \bar{x}_2 \wedge x_3 \wedge \bar{x}_4 \wedge \bar{x}_5 \wedge x_6 \wedge \bar{x}_7$
$\delta(q, e, q')$	(Running, release, Exit)	(Running, timeout, Ready)
$\tau(v, \gamma, v')$	$\bar{x}_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge \bar{x}_5 \wedge \bar{x}_6 \wedge x_7$	$\bar{x}_1 \wedge x_2 \wedge \bar{x}_3 \wedge \bar{x}_4 \wedge x_5 \wedge \bar{x}_6 \wedge \bar{x}_7$
$\delta(q, e, q')$	(Ready, waitEvent, Blocked)	(Blocked, eventOccurs, Ready)
$\tau(v, \gamma, v')$	$x_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge x_4 \wedge \bar{x}_5 \wedge x_6 \wedge x_7$	$x_1 \wedge x_2 \wedge x_3 \wedge \bar{x}_4 \wedge x_5 \wedge \bar{x}_6 \wedge \bar{x}_7$

Bảng 4.5: Thành phần $\delta(q, e, q') \mapsto \tau(v, \gamma, v')$ trong bảng ánh xạ

Cuối cùng, kết quả sau khi chuyển đổi chúng ta thu được dạng đặc tả sử dụng hàm logic $N = \langle X, E, \tau(X, E, X'), \iota(X) \rangle$ và bảng ánh xạ (Bảng 4.2, 4.3, 4.4 và 4.5). Với $N = \langle X, E, \tau(X, E, X'), \iota(X) \rangle$. Trong đó:

- $X = \{x_1, x_2\}, X' = \{x_6, x_7\},$
- $E = \{x_3, x_4, x_5\},$
- $\tau(X, E, X') = \{\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge \bar{x}_4 \wedge \bar{x}_5 \wedge \bar{x}_6 \wedge \bar{x}_7\} \mid \{x_1 \wedge \bar{x}_2 \wedge x_3 \wedge \bar{x}_4 \wedge \bar{x}_5 \wedge x_6 \wedge \bar{x}_7\}$
 $\mid \{\bar{x}_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge \bar{x}_5 \wedge \bar{x}_6 \wedge x_7\} \mid \{\bar{x}_1 \wedge x_2 \wedge \bar{x}_3 \wedge \bar{x}_4 \wedge x_5 \wedge \bar{x}_6 \wedge \bar{x}_7\} \mid$
 $\{x_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge x_4 \wedge \bar{x}_5 \wedge x_6 \wedge x_7\} \mid \{x_1 \wedge x_2 \wedge x_3 \wedge \bar{x}_4 \wedge x_5 \wedge \bar{x}_6 \wedge \bar{x}_7\},$ và
- $\iota(X) = \bar{x}_1 \wedge \bar{x}_2.$

4.3 Chuyển đổi dạng đặc tả sử dụng hàm logic sang dạng đặc tả sử dụng LTS

Giả sử chúng ta đã có dạng đặc tả sử dụng hàm logic N và bảng ánh xạ.

Với $N = \langle X, E, \tau(X, E, X'), \iota(X) \rangle$. Trong đó:

- $X = \{x_1, x_2\}, X' = \{x_6, x_7\},$
- $E = \{x_3, x_4, x_5\},$
- $\tau(X, E, X') = \{\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge \bar{x}_4 \wedge \bar{x}_5 \wedge \bar{x}_6 \wedge \bar{x}_7\} \mid \{x_1 \wedge \bar{x}_2 \wedge x_3 \wedge \bar{x}_4 \wedge \bar{x}_5 \wedge x_6 \wedge \bar{x}_7\}$
 $\mid \{\bar{x}_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge \bar{x}_5 \wedge \bar{x}_6 \wedge x_7\} \mid \{\bar{x}_1 \wedge x_2 \wedge \bar{x}_3 \wedge \bar{x}_4 \wedge x_5 \wedge \bar{x}_6 \wedge \bar{x}_7\} \mid$
 $\{x_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge x_4 \wedge \bar{x}_5 \wedge x_6 \wedge x_7\} \mid \{x_1 \wedge x_2 \wedge x_3 \wedge \bar{x}_4 \wedge x_5 \wedge \bar{x}_6 \wedge \bar{x}_7\},$ và
- $\iota(X) = \bar{x}_1 \wedge \bar{x}_2.$

Và bảng ánh xạ Map:

Q_1	New	Ready	Running	Blocked
X_1	$\bar{x}_1 \wedge \bar{x}_2$	$x_1 \wedge \bar{x}_2$	$\bar{x}_1 \wedge x_2$	$x_1 \wedge x_2$

Bảng 4.6: Thành phần $Q_1 \mapsto X_1$ trong bảng ánh xạ

Q_2	Ready	Running	Exit	Blocked
X_2	$\bar{x}_6 \wedge \bar{x}_7$	$x_6 \wedge \bar{x}_7$	$\bar{x}_6 \wedge x_7$	$x_6 \wedge x_7$

Bảng 4.7: Thành phần $Q_2 \mapsto X_2$ trong bảng ánh xạ

Nhiệm vụ của chúng ta là tìm ra LTS ban đầu, tức là LTS đã được mã hóa thành dạng đặc tả sử dụng hàm logic N đã cho. Dựa vào thành phần $Q_1 \mapsto X_1$ ta thấy $\bar{x}_1 \wedge \bar{x}_2$ là dạng mã hóa của trạng thái q_0 mà $\iota(X) = \bar{x}_1 \wedge \bar{x}_2$ nên q_0 là trạng thái khởi tạo của LTS. (*)

Lại có, dựa vào thành phần $Q_1 \mapsto X_1, Q_2 \mapsto X_2$, ta thấy Q_1 có 4 trạng thái

Σ	admit	dispatch	release
E	$\bar{x}_3 \wedge \bar{x}_4 \wedge \bar{x}_5$	$x_3 \wedge \bar{x}_4 \wedge \bar{x}_5$	$\bar{x}_3 \wedge x_4 \wedge \bar{x}_5$
Σ	timeout	waitEvent	eventOccurs
E	$x_3 \wedge x_4 \wedge \bar{x}_5$	$\bar{x}_3 \wedge \bar{x}_4 \wedge x_5$	$x_3 \wedge \bar{x}_4 \wedge x_5$

Bảng 4.8: Thành phần $\Sigma \mapsto E$ trong bảng ánh xạ

$\delta(q, e, q')$	(New, admit, Ready)	(Ready, dispatch, Running)
$\tau(v, \gamma, v')$	$\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge \bar{x}_4 \wedge \bar{x}_5 \wedge \bar{x}_6 \wedge \bar{x}_7$	$x_1 \wedge \bar{x}_2 \wedge x_3 \wedge \bar{x}_4 \wedge \bar{x}_5 \wedge x_6 \wedge \bar{x}_7$
$\delta(q, e, q')$	(Running, release, Exit)	(Running, timeout, Ready)
$\tau(v, \gamma, v')$	$\bar{x}_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge \bar{x}_5 \wedge \bar{x}_6 \wedge x_7$	$\bar{x}_1 \wedge x_2 \wedge \bar{x}_3 \wedge \bar{x}_4 \wedge x_5 \wedge \bar{x}_6 \wedge \bar{x}_7$
$\delta(q, e, q')$	(Ready, waitEvent, Blocked)	(Blocked, eventOccurs, Ready)
$\tau(v, \gamma, v')$	$x_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge x_4 \wedge \bar{x}_5 \wedge x_6 \wedge x_7$	$x_1 \wedge x_2 \wedge x_3 \wedge \bar{x}_4 \wedge x_5 \wedge \bar{x}_6 \wedge \bar{x}_7$

Bảng 4.9: Thành phần $\delta(q, e, q') \mapsto \tau(v, \gamma, v')$ trong bảng ánh xạ

là $Q_1 = \{\text{New, Ready, Running, Blocked}\}$, Q_2 có 4 trạng thái là $Q_2 = \{\text{Ready, Running, Exit, Blocked}\}$ mà $Q = Q_1 \cup Q_2$ nên $Q = \{\text{New, Ready, Running, Exit, Blocked}\}$ (**). Thêm vào đó, dựa vào thành phần $\Sigma \mapsto E$, ta thấy LTS có 6 sự kiện $\Sigma = \{\text{admit, dispatch, release, timeout, waitEvent, eventOccurs}\}$ (***). Mặt khác $\tau(X, E, X') = \{\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge \bar{x}_4 \wedge \bar{x}_5 \wedge \bar{x}_6 \wedge \bar{x}_7\} \mid \{x_1 \wedge \bar{x}_2 \wedge x_3 \wedge \bar{x}_4 \wedge \bar{x}_5 \wedge x_6 \wedge \bar{x}_7\} \mid \{\bar{x}_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge \bar{x}_5 \wedge \bar{x}_6 \wedge x_7\} \mid \{\bar{x}_1 \wedge x_2 \wedge \bar{x}_3 \wedge \bar{x}_4 \wedge x_5 \wedge \bar{x}_6 \wedge \bar{x}_7\} \mid \{x_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge x_4 \wedge \bar{x}_5 \wedge x_6 \wedge x_7\} \mid \{x_1 \wedge x_2 \wedge x_3 \wedge \bar{x}_4 \wedge x_5 \wedge \bar{x}_6 \wedge \bar{x}_7\}$. Xét lần lượt từng phần tử trong $\tau(X, E, X')$, với $\tau(v, \gamma, v') = \{\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge \bar{x}_4 \wedge \bar{x}_5 \wedge \bar{x}_6 \wedge \bar{x}_7\}$ (kể cả khi thứ tự các biến mã hóa bị thay đổi). Đối chiếu với thành phần $\Sigma \mapsto E$ của bảng ánh xạ ta thấy đây là dạng biểu diễn của sự kiện admit. Một các hoàn toàn toàn tương tự ta sẽ có các chuyển trạng thái còn lại trong LTS là (Ready, dispatch, Running), (Running, release, Exit), (Running, timeout, Ready), (Running, waitEvent, Blocked) và (Blocked, eventOccurs, Ready). (****)

Từ (*) (**) (***) và (****) ta được LTS M như hình 4.1. $M = \langle Q, \Sigma, \delta, q_0 \rangle$

Trong đó:

- $Q = \{\text{New, Ready, Running, Exit, Blocked}\}$, $Q_1 = \{\text{New, Ready, Running, Blocked}\}$, $Q_2 = \{\text{Ready, Running, Exit, Blocked}\}$,
- $\Sigma = \{\text{admit, dispatch, timeout, release, waitEvent, eventOccurs}\}$,
- $\delta = \{(\text{New, admit, Ready}), (\text{Ready, dispatch, Running}), (\text{Running, release, Exit}), (\text{Running, timeout, Ready}), (\text{Ready, waitEvent, Blocked}), (\text{Blocked, eventOccurs, Ready})\}$, và

- New là trạng thái khởi tạo.

Tài liệu tham khảo

- [1] Yu-Fang Chen, Edmund M. Clarke, Azadeh Farzan, Ming-Hsien Tsai, Yih-Kuen Tsay, and Bow-Yaw Wang, *Automated Assume-Guarantee Reasoning through Implicit Learning*. Addison-Wesley, Reading, Massachusetts, 1993.
- [2] Pham Ngoc Hung, *Assume-Guarantee Verification of Evolving Component-Based Software*. Japan Advanced Institute of Science and Technology in partial fulfillment of the requirements for the degree of Doctor of Philosophy, September, 2009.
- [3] P. N. Hung, N. V. Ha, T. Aoki and T. Katayama, *On Optimization of Minimized Assumption Generation Method for Component-based Software Verification* IEICE Trans. on Fundamentals, Special Issue on Software Reliability Engineering, Vol. E95-A, No.9, pp. 1451-1460, Sep. 2012.