Lu^{a}n v«n th ¤c s¾

Ch÷ìng 1

Giîi thi»u

2

Ch÷ìng 2

Ki¸n thùc cì sð

2.1 D¤ng °c t£ sû döng h» chuy°n tr¤ng th¡i ÷ñ c g-n nh¢n

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 = $< Q, \Sigma, \delta, q_0 >$
Trong đó:

- Q = $\{q_0, q_1, ..., q_n\}$ là tập các trạng thái,

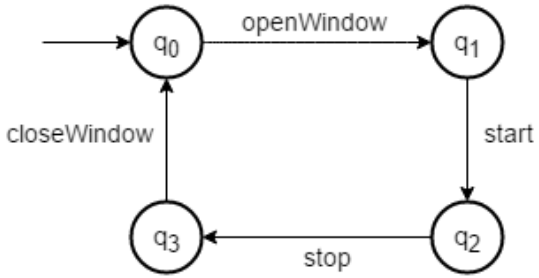
- $\Sigma = \{w_0, w_1, ..., w_n\}$ là tập các sự kiện,
- $\delta \subseteq$ Q x Σ x Q là hàm chuyển trạng thái, và
- $q_0 \subseteq$ Q là trạng thái khởi tạo.

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 $= <$ Q, $\Sigma, \delta, q_0 >$, trong đó:

- Q $= \{q_0, q_1, q_2, q_3\}$,



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

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

Định nghĩa 2.2: Kích thước của một tập hợp [1].
Kích thước của một tập hợp $Q = \{q_0, q_1, ..., q_n\}$ 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 $M = < Q, \Sigma, \delta, q_0 >$ 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 $M = < Q, \Sigma, \delta, q_0 >$ là một chuỗi hữu hạn các sự kiện có dạng

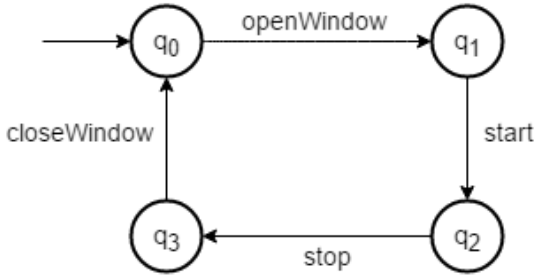
Vết của một LTS $M = < Q, \Sigma, \delta, q_0 >$ là một chuỗi hữu hạn các sự kiện có dạng $\sigma = w_0 w_1 ... w_k$ với $w_k \in \Sigma$ và $0 \leq k \leq n$ sao cho tồn tại trạng thái $q_i \in Q$ mà $q_0 \xrightarrow{\sigma} q_i$.

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 $M = < Q, \Sigma, \delta, q_0 >$, 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 2.2: Minh họa vết của LTS.

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 thông 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.

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) = $\{\alpha \mid \alpha$ 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) = $\{openWindow, onpenWindow\ start, openWindow\ start\ stop, ...\}$

2.2 D¤ng °c t£ sû dông h m logic (Bo ol)

Định nghĩa 2.6: Hàm logic [1]

Định nghĩa 2.5: Hàm logic [1].

B = {T, F} là miền giá trị logic. Với X là tập hợp các biến logic, một hàm logic θ(X) được định nghĩa θ(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 θ(X): $B^3 \rightarrow B$.

Định nghĩa 2.7: Phép gán [1].

Với X là tập hợp các biến logic, phép gán υ được đinh nghĩa υ: 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, $\upsilon_1(x) =$ T, $\upsilon_2(x) =$ F, $\upsilon_1(y) =$ T, $\upsilon_2(y) =$ F, $\upsilon_1(z) =$ T và $\upsilon_2(z) =$ 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, υ là một phép gán trên tập X, phép gán hàm kí hiệu $\phi[\upsilon]$ là kết quả thu được khi thay các phần tử $x \in X$ bởi $\upsilon(x)$. 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 $\upsilon(x)$ và $\upsilon'(x')$ lần lượt là các phép gán trên tập X và X′, kí hiệu $\psi[\upsilon, \upsilon']$ là kết quả thu được khi thay một cách tương ứng các phần tử $x \in X$ bởi $\upsilon(x)$ và $x' \in X'$ bởi $\upsilon'(x')$.

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

Một cách tổng quát, với n tập các biến logic X, X_1, X_2, ..., X_n trong đó $X_i = \{x_i \mid x \in X\}$, $\psi(X, X_1, X_2, ..., X_n)$ là hàm logic tương ứng trên các tập biến logic X, X_1, X_2, ..., X_n, ta kí hiệu $\psi[\upsilon_1, \upsilon_2, ..., \upsilon_n]$ là kết quả thu được khi thay một cách tương ứng các phần tử $x_1 \in X_1$ bởi $\upsilon_1(x_1)$, $x_2 \in X_2$ bởi $\upsilon_2(x_2)$, ... và $x_n \in X_n$ bởi $\upsilon_n(x_n)$.

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:

$N = < X, E, \tau(X, X'), \iota(X) >$

Trong đó:

- X là tập các biến logic dùng để biểu diễn các trạng thái của hệ thống. $X = \{x_0, x_1, ..., x_n\}$,
- E là tập các biến logic dùng để biểu diễn các hành vi của hệ thống. $E = \{e_0, e_1, ..., e_n\}$,
- $\tau(X, X')$ 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 = < X, E, $\tau(X, X')$, ι(X) >. Trong đó:

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

Định nghĩa 2.10: Vết của dạng đặc tả sử dụng hàm logic.
Với dạng đặc tả sử dụng hàm logic N = <X, E, τ, ι(X)>, υ 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 $\xi = \gamma^0\gamma^1...\gamma^n$ được gọi là vết của N khi và chỉ khi tồn tại tập các phép gán υ^0, υ^1, ..., υ^{n+1}, γ^0, γ^0, ..., γ^n sao cho $\iota[\upsilon^0]$ = T và $\tau[\upsilon^i, \gamma^i, \upsilon^{i+1}]$ = T với 0 ≤ i ≤ 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 = <X, E, τ(X, X′), ι(X)>. Trong đó:

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

Vì $\iota(X) = \overline{x}_1 \wedge \overline{x}_2$, với υ^0 là phép gán trên tập X sao cho $\upsilon^0(x_1)$ = F và $\upsilon^0(x_2)$ = F nên $\iota[\upsilon^0]$ = T ∧ T = T. Mặt khác, gọi υ^1 là phép gán trên tập X sao cho $\upsilon^1(x_5)$ = T, $\upsilon^1(x_6)$ = F, γ^0 là phép gán trên tập E sao cho $\gamma^0(x_3)$ = F và $\gamma^0(x_4)$ = F. Khi đó, $\tau[\upsilon^0, \gamma^0, \upsilon^1]$ = T nên $\xi = \gamma^0$ là một vết của N. Mặt khác, với phép gán υ^1, γ^1, υ^2 sao cho $\upsilon^1(x_1)$ = T, $\upsilon^1(x_2)$ = F, $\gamma^1(x_3)$ = T, $\gamma^1(x_4)$ = F, $\upsilon^2(x_5)$ = F, $\upsilon^2(x_6)$ = T thì $\tau[\upsilon^1, \gamma^1, \upsilon^2]$ = 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 = <X, E, τ, ι(X)>, 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) = { ξ | ξ 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) = $\{\overline{x}_3 \wedge \overline{x}_4, \overline{x}_3 \wedge \overline{x}_4 \wedge x_3 \wedge \overline{x}_4, \overline{x}_3 \wedge \overline{x}_4 \wedge x_3 \wedge \overline{x}_4 \wedge \overline{x}_3 \wedge x_4, ...\}$

2.3 B£ng ¡nh x¤

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 $M = < Q, \Sigma, \delta, q_0 >$ 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 $N = < X, E, \tau, \iota(X) >$, trong đó $X = X_1 \cup X_2$ với X_1 là tập các biến logic dùng để biểu diễn các trang 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:

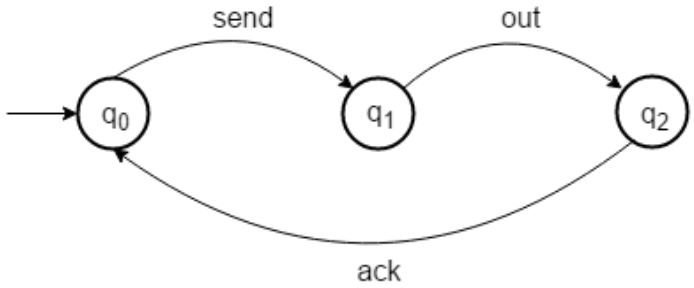
$$\begin{cases} Q_1 \mapsto X_1 \end{cases}$$

$$Map = \begin{cases} \mathrm{Q}_2 \mapsto \mathrm{X}_2 \\ \Sigma \mapsto E \\ \delta \mapsto \tau \\ q_0 \mapsto \iota \end{cases}$$

Trong đó:

- $\mathrm{Q}_1 \mapsto \mathrm{X}_1$ là một song ánh từ tập các trạng thái đầu vào đến tập tập các biến logic dùng để biểu diễn các trạng thái đầu vào của hệ thống,
- $\mathrm{Q}_2 \mapsto \mathrm{X}_2$ là một song ánh từ tập các trạng thái đầu ra đến tập 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,
- $\delta \mapsto \tau$ 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,
- $q_0 \mapsto \iota$ là một ánh xạ từ trạng thái khởi tạo đến dạng mã hóa trạng thái khởi tạo của hệ thống.

Ví dụ 2.12: Ví dụ về bảng ánh xạ.
Cho LTS $\mathrm{M} = < \mathrm{Q}, \Sigma, \delta, q_0 >$ 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 $N = < X, E, \tau(X, E, X'), \iota(X) >$. Trong đó:

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

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

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	$\overline{x}_1 \wedge \overline{x}_2$	$x_1 \wedge \overline{x}_2$	$\overline{x}_1 \wedge x_2$

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

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

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

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

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

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

Bảng 2.4: Thành phần $\delta \mapsto \tau$ trong bảng ánh xạ

q_0	q_0
ι	$\overline{x}_1 \wedge \overline{x}_2$

Bảng 2.5: Thành phần $q_0 \mapsto \iota$ trong bảng ánh xạ

Ch÷ìng 3

Ph÷ìng ph¡p chuy°n êi

3.1 Th u ªt to¡n

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 Th u ªt to¡n m¢ hâa mët t ªp hñp

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 hõa 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

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

```
for mỗi phần tử a_i trong tập A do
    Lưu a_i vào trong bảng ánh xạ
    α_i = True
    k = Thứ tự của a_i trong A
```

```
5   Chuyển k sang số nhị phân với độ dài [log2(|A|)] + 1 bit
6   for mỗi bit trong chuỗi nhị phân biểu diễn k do
7       if bit = 0 then
8           Biểu diễn bit dưới dạng x̄_j
9       else
10          Biểu diễn bit dưới dạng x_j
11      end
12      α_i = α_i ∧ x_j
13  end
14  Lưu α_i vào bảng ánh xạ ứng với vị trí của phần tử a_i
15 end
```

Với A là tập các phần tử cần mã hóa A = $\{a_0, a_1, ..., a_n\}$. Số biến logic cần dùng để mã hóa tập hợp A là m = $[log_2 n] + 1$. Gọi X = $\{x_1, x_2, ..., x_m\}$ 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 $x_1 \wedge x_2 \wedge ... \wedge x_m$. Xét một phần tử bất kì a_i trong tập A, theo bước (4) ta xác định được thứ tự của a_i trong tập A là k, theo bước (5) thì k sẽ biểu diễn dưới dạng số nhị phân m = $[log_2 n] + 1$ bit. Theo bước (6)-(12) chúng ta sẽ mã hóa được phần tử a_i, kết quả sau khi vòng lặp ở bước (6) kết thúc chúng ta sẽ có được dạng mã hóa của phần tử a_i. Thêm vào đó, theo bước (14) 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 $\alpha_i = x \bigwedge_{i=1}^{n} x_i$ là một hàm logic biểu diễn a_i và các phần tử x_i biểu diễn hàm α_i đều thuộc tập X_i với $X_i = \{x_i \mid x \subset X\}$

biểu diễn hàm α_i đều thuộc tập X_i với $X_i = \{x_i \mid x \in X\}$

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 Th uªt to¡n m¢ hâa tªp c¡c truy ºn tr¤ng th¡i

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 $\tau(\mathrm{X}, \mathrm{E}, \mathrm{X}')$

1 **for** *mỗi chuyển trạng thái được biểu diễn* $\delta(q, w, q')$ **do**

2 Lấy dạng biểu diễn α_i của q từ bảng ánh xạ của tập các trạng thái đầu vào

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 α_{i+1} của q' từ bảng ánh xạ của tập các trạng thái đầu ra

5 Biểu diễn chuyển trạng thái dưới dạng: $\alpha_i \wedge e_i \wedge \alpha_{i+1}$

6 **end**

7 **return** $\tau = \bigvee_{i=1}^{n} \{\alpha_i \wedge e_i \wedge \alpha_{i+1}\}$

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_1, 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 Chùng 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

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 = <Q, Σ, δ, q_0 >, L(M) là ngôn ngữ của của M và một dạng đặc tả sử dụng hàm logic N = < X, E, τ(X, X'), ι(X) >, L(N) là ngôn ngữ của

N, N được chuyển đổi từ M. Với $\alpha = w_0 w_1 ... w_n$ là vết của LTS M, áp dụng thuật toán 3.1 ta thu được $\xi = \gamma^0 \gamma^1 ... \gamma^n$, 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 υ^0 cho hàm α_0 trên tập X sao cho $\iota[\upsilon^0] = \mathrm{T}$. (*)

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 một chuỗi hội của các biến logic nên tồn tại các phép gán υ^i cho hàm α_i trên tập X, υ^{i+1} cho hàm α_{i+1} trên tập X', γ^i là phép gán cho hàm e_i trên tập E sao cho $\upsilon^i = \mathrm{T}$, $\upsilon^{i+1} = \mathrm{T}$, $\gamma^i = \mathrm{T}$ để $\tau[\upsilon^i, \gamma^i, \upsilon^{i+1}] = \mathrm{T}$ (**). Từ (*), (**) và định nghĩa 2.10, $\xi = \gamma^0 \gamma^1 ... \gamma^n$ là một vết của N hay $\xi \in \mathrm{L(N)}$.

Chứng minh mệnh đề 2:

Với một LTS $\mathrm{M} = <\mathrm{Q}, \Sigma, \delta, \mathrm{q}_0>$, L(M) là ngôn ngữ của của M và một dạng đặc tả sử dụng hàm logic $\mathrm{N} = <\mathrm{X}, \mathrm{E}, \tau(\mathrm{X}, \mathrm{X}'), \iota(\mathrm{X})>$, 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 $\xi = \gamma^0 \gamma^1 ... \gamma^n$ 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 $\xi = \gamma^0 \gamma^1 ... \gamma^n$ là một vết của N nên tồn tại các phép gán υ^0, υ^1, ..., υ^{n+1}, γ^0, γ^1, ..., γ^n sao cho $\iota[\upsilon^0] = \mathrm{T}$ và $\tau[\upsilon^i, \gamma^i, \upsilon^{i+1}] = \mathrm{T}$ với $\forall i$: $0 \le i \le n$ mà υ^0 là phép gán của α_0 trên tập X, dựa vào bảng ánh xạ α_0 tương ứng với q_0 nên q_0 là trạng thái khởi tạo của M. Thêm vào đó, vì $\tau[\upsilon^i, \gamma^i, \upsilon^{i+1}] = \mathrm{T}$ nên dựa vào bảng ánh xạ Map ta thu được hàm chuyển trạng thái $\delta(q_i, w_i, q_{i+1})$ tương ứng. Vì $0 \le i \le n$ nên ta có các chuyển trạng thái $\delta(q_0, w_0, q_1), \delta(q_1, w_1, q_2), ... \delta(q_i, w_i, q_{i+1})$, ứng với các chuyển trạng thái này ta được chuỗi các sự kiện $\sigma = w_0 w_1 ... w_i$ 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_{n-1}, sau khi sự kiện w_{n-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 $\forall i$: $0 \leq i < n$, tồn tại trạng thái $q_i \in$ Q sao cho $q_0 \xrightarrow{\sigma} q_i$ nên theo định nghĩa 2.4, $\sigma = w_0 w_1 ... w_n$ là một vết của M.

Ch÷ìng 4

V½dö v· vi»c chuyºn êi giúa c¡c d¤ng °c t£

T i li»u tham kh£o

[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.