Tạp chí Khoa học ĐHQGHN, Khoa học Tự nhiên và Công nghệ 27 (2011) 225-232

Đặc tả và kiểm chứng tính bất biến của các hệ đa tác tử

Phạm Ngọc Hùng[1,*], Đào Anh Hiển[2], Nguyễn Ánh Nguyệt[1], Nguyễn Việt Hà[1]

[1]*Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội, 144 Xuân Thủy, Hà Nội, Việt Nam*
[2]*Trường Đại học Sư Phạm Kỹ Thuật Hưng Yên, Dân Tiến, Khoái Châu, Hưng Yên*

Nhận ngày 10 tháng 8 năm 2011

Tóm tắt. Chứng minh tính đúng đắn của các hệ thống nói chung và các hệ đa tác tử nói riêng đang nhận được sự quan tâm nghiên cứu rộng rãi. Bài toán này sẽ khó khăn hơn khi các hệ đa tác tử có không gian trạng thái là vô hạn. Bài báo này đề xuất một phương pháp đặc tả và kiểm chứng các thuộc tính bất biến của các hệ đa tác tử đối với không gian trạng thái là vô hạn. Trong phương pháp này, hành vi của hệ thống cũng với các thuộc tính cần chứng minh được đặc tả bằng ngôn ngữ đại số. Không gian vô hạn các trạng thái của hệ thống được xác định đệ quy bằng cách chỉ ra trạng thái khởi tạo và cách chuyển đến các trạng thái tiếp theo từ một trạng thái bất kỳ của hệ thống. Chúng tôi sử dụng phương pháp quy nạp toán học để chứng minh tính thỏa mãn của các thuộc tính trên toàn bộ không gian trạng thái của hệ thống. Một ví dụ minh họa cho cũng được trình bày và thảo luận trong bài báo này nhằm minh chứng cho tính hiệu quả của phương pháp đề xuất.

Từ khóa: specification and verification, multi-agent systems, invariant properties, CafeOBJ, induction proof.

1. Giới thiệu

Các hệ đa tác tử đang được quan tâm nghiên cứu và ứng dụng ngày càng rộng rãi [1-4]. Trong công nghệ phần mềm, các hệ đa tác tử được quan tâm như là một mô hình phát triển phần mềm trong tương lai, đặc biệt là cho các phần mềm có quy mô lớn nơi mà các thành phần phần mềm có tính độc lập và tự trị như các tác tử [4]. Tuy nhiên, cũng như các hệ thống nói chung, việc đảm bảo tính đúng đắn của các hệ đa tác tử (làm thế nào để đảm bảo rằng các tác tử sẽ kết hợp với nhau để đạt được mục tiêu của hệ thống) là một trong những vấn đề chưa có giải pháp hiệu quả và đang được quan tâm nghiên cứu. Giải pháp phổ biến hiện nay là áp dụng các kỹ thuật kiểm thử (testing). Tuy nhiên, kiểm thử chỉ có khả năng phát hiện ra lỗi/khiếm khuyết của hệ thống chứ không chỉ ra được hệ thống không còn lỗi. Vì lý do này mà với các hệ thống yêu cầu độ tin cậy cao thì kiểm thử là không đủ để đảm bảo chất lượng hệ thống. Một trong những giải pháp giải quyết vấn đề này là áp dụng các phương pháp kiểm chứng mô hình [5-7]. Tuy nhiên, các phương pháp này chỉ áp dụng được khi số lượng các tác tử của hệ thống là hữu hạn. Hơn nữa, vấn đề bùng nổ không gian trạng thái có thể xảy ra khi áp dụng các phương pháp kiểm chứng mô hình cho các hệ thống lớn. Hơn nữa, trong thực tế, số lượng tác tử trong các hệ đa tác tử thường là

* Tác giả liên hệ. ĐT: 84-4-37549016.
E-mail: hungpn@vnu.edu.vn

chưa biết trước vì nó thường xuyên thay đổi trong quá trình phát triển và thậm chí là trong quá trình thực thi hệ thống. Điều này dẫn đến việc khó có thể áp dụng các phương pháp kiểm chứng mô hình nhằm chứng minh tính đúng đắn của hệ thống.

Bài báo này đề xuất một phương pháp đặc tả và kiểm chứng các thuộc tính bất biến của các hệ đa tác tử nhằm giải quyết những vấn đề trên. Trong phương pháp đề xuất, chúng tôi sử dụng ngôn ngữ đại số để đặc tả hành vi của các tác tử. Từ những đặc tả này, chúng tôi sẽ xây dựng không gian trạng thái của hệ thống một cách đệ quy gồm trạng thái khởi tạo và cách chuyển đến các trạng thái tiếp theo từ một trạng thái bất kỳ của hệ thống. Phương pháp đề xuất cho phép đặc tả và chứng minh tính đúng đắn của các hệ đa tác tử với không gian trạng thái là vô hạn. Chúng tôi cũng áp dụng phương pháp đề xuất để chứng minh vấn đề xung đột tài nguyên không được phép xảy ra trong một hệ đa tác tử nhằm chỉ ra tính hiệu quả của phương pháp này. Trong ví dụ minh họa này, chúng tôi sử dụng bộ chứng minh định lý CafeOBJ [8,9] để hiện thực hóa phương pháp đặc tả và kiểm chứng đề xuất.

Phần còn lại của bài báo được cấu trúc như sau. Phần 2 trình bày phương pháp đặc tả các hệ đa tác tử sử dụng ngôn ngữ đại số. Phương pháp kiểm chứng tính đúng đắn của các hệ đa tác tử đối với các thuộc tính bất biến sử dụng tư tưởng quy nạp toán học được trình bày trong phần 3. Phần 4 trình bày một ví dụ minh họa nhằm minh chứng cho tính hiệu quả của phương pháp đề xuất và thảo luận kết quả thu được. Cuối cùng, kết luận của bài báo và hướng nghiên cứu tiếp theo được trình bày ở phần 5.

2. Đặc tả hệ thống đa tác tử

Để kiểm chứng rằng liệu các tác tử có kết hợp được với nhau để đạt được mục tiêu của hệ thống, chúng ta phải đặc tả hành vi của hệ thống thông qua hành vi của từng tác tử. Các thuộc tính cần kiểm chứng cũng phải được đặc tả. Trong nghiên cứu này, chúng tôi chỉ quan tâm các thuộc tính bất biến. Các thuộc tính này yêu cầu phải được thỏa mãn tại mọi trạng thái của hệ thống.

2.1. Đặc tả tác tử

Gọi AId là tập các chỉ số của các tác tử và Sys là không gian trạng thái của hệ thống đa tác tử, với mỗi tác tử $i \in$ AId, tập hữu hạn các hành động a_{i1}, a_{i2}, …, a_{in} của nó được định nghĩa như sau: a_{ij}: AId × Sys → Sys với j = 1, …, n.

Với $s \in$ Sys và con_a_{ij}: AId × Sys → {true, false}, $a_{ij}(i, s) = s'$ ($s' \in$ Sys, $s' \neq s$) nếu $con\_a_{ij}(i, s)$ = true, ngược lại, $a_{ij}(i, s) = s$. Trong trường hợp $s' = a_{ij}(i, s)$ ($s' \neq s$), ta nói s' là trạng thái tiếp theo của s khi tác tử i thực hiện thành công hành động a_{ij} tại trạng thái này.

2.2. Không gian trạng thái

Không gian trạng thái của hệ thống đa tác tử (ký hiệu là Sys) là tập vô hạn các trạng thái bắt đầu từ trạng thái khởi tạo của hệ thống (ký hiệu là init). Tại mỗi trạng thái $s \in$ Sys, hệ thống sẽ chuyển đến một trạng thái tiếp theo $s' = a_{ij}(i, s)$ nếu một trong các tác tử $i \in$ AId thực hiện thành công hành động a_{ij} tại trạng thái s. Không gian trạng thái của hệ thống đa tác tử được định nghĩa đệ quy như hình 1.

$$Sys = \{init\} \cup \{a_{ij}(i, s) \mid i \in AId, s \in Sys, j \in [1..n]\}$$

Hình 1. Định nghĩa đệ quy không gian trạng thái của hệ đa tác tử.

2.3. Đặc tả các thuộc tính bất biến

Có một số thuộc tính của hệ đa tác tử cần kiểm chứng trước khi triển khai chúng trong thực tế. Trong bài báo này, chúng tôi tập trung giải quyết thuộc tính bất biến – một thuộc tính phổ biến của các hệ thống nói chung và các hệ đa tác tử nói riêng. Thuộc tính bất biến là tính chất mà hệ thống phải thỏa mãn tại mọi trạng thái của hệ thống. Một thuộc tính bất biến được định nghĩa như sau: inv: Sys $\rightarrow$ {true, false}. Chúng ta cần kiểm tra rằng $\forall s \in$ Sys thì inv(s) = true.

3. Kiểm chứng hệ thống đa tác tử

Phương pháp đang được áp dụng phổ biến hiện nay để chứng minh tính đúng đắn của các hệ đa tác tử là kiểm chứng mô hình [5-7]. Tuy nhiên, một trong những hạn chế lớn nhất của kiểm chứng mô hình là vấn đề bùng nổ không gian trạng thái khi áp dụng cho các hệ thống lớn [6]. Vì lý do này mà phương pháp này khó áp dụng trong thực tế. Hơn nữa, phương pháp kiểm chứng mô hình chỉ áp dụng cho các hệ thống có không gian trạng thái là hữu hạn. Các hệ thống đa tác tử với số lượng tác tử là không biết trước hoặc vô hạn thì không gian trạng thái của hệ thống có thể là vô hạn. Điều này có nghĩa là chúng ta không thể áp dụng phương pháp kiểm chứng mô hình trong trường hợp này.

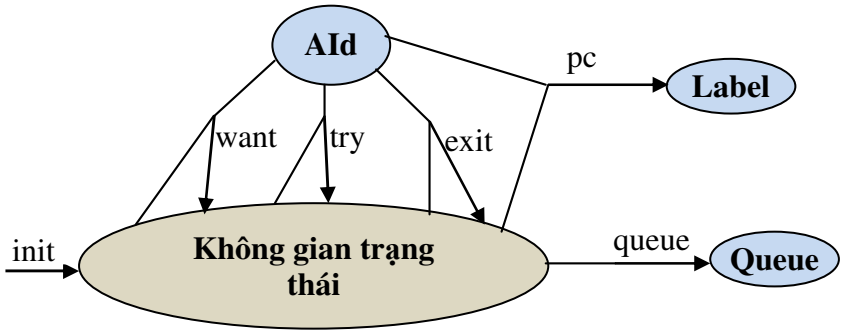
Để chứng minh tính đúng đắn của các hệ đa tác tử, chúng tôi đề xuất sử dụng phương pháp chứng minh theo tư tưởng của quy nạp toán học. Giả sử chúng ta cần chứng minh thuộc tính inv đúng trên mọi trạng thái của hệ thống, quy trình chứng minh được mô tả trong hình 2. Tại trường hợp cơ sở, chúng ta kiểm tra thuộc tính inv có thỏa mãn tại trạng thái khởi tạo hay không (inv(init) = true?). Nếu đúng thì chuyển sang bước chứng minh quy nạp, nếu sai thì hệ thống không thỏa mãn thuộc tính inv. Tại bước chứng minh quy nạp, giả sử thuộc tính inv đúng tại một trạng thái s $\in$ Sys (inv(s) = true). Ta cần chứng minh inv cũng đúng tại tất cả các trạng thái tiếp theo của s. Các trạng thái tiếp theo của s là các trạng thái của hệ thống thu được bằng cách một tác tử bất kỳ thực hiện một hành động của nó tại trạng thái s. Nếu thuộc tính inv thỏa mãn tại tất cả các trạng thái tiếp theo của s thì hệ thống thỏa mãn inv. Ngược lại, hệ thống không thỏa mãn inv. Tuy nhiên, trong quá trình chứng minh tính đúng đắn của inv tại mỗi trạng thái tiếp theo của s, có nhiều trường hợp kết quả thu được là một biểu thức logic không phải giá trị true hoặc false. Trong những trường hợp này, chúng ta cần cung cấp thêm các tiên đề hoặc hệ quả để hệ thống có thể chứng minh tính thỏa mãn của thuộc tính này. Các hệ quả này dựa trên các tính chất của hệ thống liên quan đến thuộc tính inv. Trước khi cung cấp các hệ quả này, chúng cần được chứng minh tính đúng đắn như các thuộc tính riêng biệt của hệ thống.

```
Base case: inv(init) = true?
Induction case:
{inv(s) = true | s ∈Sys} ->
                          {inv(a_ij(i, s) = true | ∀i∈AId, ∀j∈[1.. n]}?
```

Hình 2. Quy trình kiểm chứng tính bất biến của hệ đa tác tử.

4. Ví dụ minh họa

Xét một hệ thống gồm các tác tử sao cho trong quá trình thực hiện, các tác tử phải truy cập đến một tài nguyên dùng chung duy nhất của hệ thống. Chúng ta phải đảm bảo rằng tài nguyên dùng chung chỉ được sử dụng bởi một tác tử tại mỗi thời điểm – vấn đề xung đột tài nguyên. Chúng ta sử dụng hàng đợi để lưu trữ các yêu cầu truy cập tài nguyên của các tác tử. Tại trạng thái khởi tạo, các tác tử chưa có yêu cầu sử dụng tài nguyên nên chúng được gán nhãn là rm (chưa có nhu cầu sử dụng tài nguyên). Khi một tác tử nào đó có yêu cầu sử dụng tài nguyên thì nó thực hiện hành động want và nhãn mới của nó là wt (đang đợi để được sử dụng tài nguyên). Trong trường hợp này, hệ thống sẽ chuyển sang một trạng thái mới và chỉ số của tác tử sẽ được đẩy vào hàng đợi. Nếu tác tử đang ở đầu hàng đợi thì nó sẽ được sử dụng tài nguyên và nhãn mới của nó là cs (đang sử dụng tài nguyên). Ngược lại, sau một thời gian xác định nó sẽ thực hiện hành động try để cố gắng sử dụng tài nguyên dùng chung. Khi một tác tử đang sử dụng tài nguyên dùng chung và muốn kết thúc việc sử dụng tài nguyên thì nó thực hiện hành động exit và nhãn mới của nó là rm. Cơ chế hoạt động của hệ thống được mô tả trong hình 3. AId là miền giá trị chứa các id của các tác tử, Label là miền chứa các nhãn của các tác tử và Queue là miền chưa các hàng đợi được sinh ra trong quá trình hoạt động của hệ thống. Hàm pc được sử dụng để xác định nhãn của một tác tử ứng với mỗi trạng thái của hệ thống trong khi hàm queue xác định giá trị của hàng đợi tại mỗi trạng thái của hệ thống.



Hình 3. Cơ chế hoạt động của một hệ đa tác tử.

Chúng tôi sử dụng ngôn ngữ CafeOBJ [8] để hiện thực hóa phương pháp đặc tả sử dụng ngôn ngữ đại số đã trình bày trong phần 3. Hình 4 mô tả đặc tả hình thức của hệ đa tác tử với cơ chế hoạt động như đã mô tả ở trên. Điểm đặc biệt ở phương pháp đặc tả này là các hành động của mỗi tác tử tại mỗi trạng thái của hệ thống có thể được thực hiện hoặc không. Ví dụ, khi một tác tử muốn sử dụng tài nguyên bằng cách thực hiện hành động want nhưng nếu trạng thái hiện tại chỉ ra rằng tác tử này đã gửi yêu cầu rồi thì hành động này không được thực hiện và hệ thống không chuyển sang trạng thái mới sau hành động này.

```
-- Sys duoc dung de dai dien cho Khong gian trang thai cua HT
*[Sys]*
-- Khai bao init la trang thai khoi tao
op init -> Sys
-- Khai bao cac Kieu Queue, AId va Label
[Queue AId Label]
-- Khai bao cac ham pc và queue
bop pc : Sys AId -> Label
bop queue : Sys -> Queue
-- Khai bao cac hanh dong cua cac tac tu
bop want : Sys AId -> Sys
bop try : Sys AId -> Sys
bop exit : Sys AId -> Sys
```

Hình 4. Đặc tả hình thức của hệ thống bằng ngôn ngữ CafeOBJ.

Cùng với đặc tả của hệ thống, thuộc tính cần chứng minh (vấn đề xung đột tài nguyên) được đặc tả trong hình 5. Thuộc tính inv là một hàm với đầu vào là một trạng thái của hệ thống và hai chỉ số của hai tác tử của hệ thống. Hàm sẽ trả về true nếu vẫn đề xung đột tài nguyên không xảy ra tại trạng thái này và trả về fasle nếu ngược lại. Ý nghĩa của khai náo này là tại mọi trạng thái của hệ thống nếu có hai tác tử I và J cũng dùng tài nguyên thì chúng là một.

```
-- Khai bao ham inv su dung de dac ta thuoc tinh xung dot tai nguyen
op inv : Sys AId AId -> Bool
-- Khai bao cac bien
var S :  Sys
vars I J : AId
-- Dinh nghia y nghia cua thuoc tinh inv
eq inv(S,I,J) = (((pc(S,I) = cs) and (pc(S,J) = cs)) implies I = J).
```

Hình 5. Đặc tả hình thức thuộc tính xung đột tài nguyên.

Chúng tôi tiến hành chứng minh liệu đặc tả của hệ thống có thỏa mãn thuộc tính inv hay không bằng phương pháp kiểm chứng như đã mô tả ở phần 3. Hình 6 mô tả các trường hợp mà chung tôi đã xem xét để chứng minh tính thỏa mãn thuộc tính inv của hệ thống.

```
1.  init
2.  want(s,k), c-want(s,k), i = k
3.  want(s,k), c-want(s,k), ~(i = k), j = k
4.  want(s,k), c-want(s,k), ~(i=k), ~(j=k), inv(s,i,j)->inv(s',i,j)
5.  want(s,k), ~c-want(s,k)
6.  try(s,k), c-try(s,k), i = k, j = k
7.  try(s,k), c-try(s,k), i = k, ~(j = k), lemma1
8.  try(s,k), c-try(s,k), ~(i = k), j = k, lemma1
9.  try(s,k), c-try(s,k), ~(i = k), ~(j = k)
10. try(s,k), ~c-try(s,k)
11. exit(s,k), c-exit(s,k), i = k
12. exit(s,k), c-exit(s,k), ~(i = k), j = k
13. exit(s,k), c-exit(s,k), ~(i = k), ~(j = k)
14. exit(s,k), ~c-exit(s,k)
```

Hình 6. Các trường hợp cần xem xét khi chứng minh thuộc tính inv.

Tại trường hợp cơ sở, chúng tôi kiểm tra giá trị của inv(init, i, j) với mọi i và j bất kỳ và thu được kết quả true. Điều này có nghĩa là thuộc tính inv đúng tại trạng thái khởi tạo init. Giả sử thuộc tính inv đúng tại một trạng thái s bất kỳ (inv(s,i,j) = true), chúng ta cần chứng minh inv đúng tại tất cả các trạng thái tiếp theo của s. Các trạng thái tiếp theo của s là want(s,k), try(s,k) và exit(s,k). Hệ thống sẽ chuyển từ trạng thái s đến một trong các trạng thái này khi một tác tử k nào đó thực hiện một trong ba hành động của nó. Với mỗi trạng thái tiếp theo, thuộc tính inv không thỏa mãn ngay mà chúng tôi phải xem xét rất nhiều trường hợp con tương đương của nó. Ví dụ trong trường hợp 4, sau khi đưa hết tất cả các thông tin cần thiết nhưng inv(s',i,j) vẫn không trả lại giá trị true. Trong trường hợp này, biểu thức inv(s, i, j)->inv(s', i, j) (s' = want(s, k)) được sử dụng và trả lại giá trị true. Về mặt ngữ nghĩa, biểu thức này tương đương với inv(s',i,j) vì chúng ta đã giả sử inv đúng tại s. Bằng phương pháp này, chúng tôi đã chứng minh được rằng inv(s', i, j) = true với s' = want(s, k). Các trường hợp còn lại cũng được chứng minh tương tự. Tuy nhiên, trong một số trường hợp chúng ta phải cung cấp thêm một số bổ đề (ví dụ như các trường hợp 7 và 8). Các bổ đề này được xác định dựa vào tính chất của hệ thống và cần được chứng minh trước khi đưa vào áp dụng.

Ví dụ này là minh chứng cho khả năng chứng minh các thuộc tính bất biến của các hệ đa tác tử với không gian trạng thái là vô hạn. Với cách tiếp cận này, chúng ta không cần chỉ ra tất cả các trạng thái của hệ thống mà chỉ cần chỉ ra trạng thái khởi tạo và quy luật chuyển đến các trạng thái tiếp theo của một trạng thái bất kỳ của hệ thống. Tuy nhiên, phương pháp chứng minh quy nạp là bán tự động và đòi hỏi người áp dụng phải có hiểu biết xâu sắc về quy nạp toán học cũng như những tính chất của hệ thống. Trong nhiều trường hợp, chúng ta phải bổ sung các bổ đề nhằm cung cấp thêm tri thức cho công cụ chứng minh để kết luận tính đúng đắn của thuộc tính tại mỗi trạng thái. Việc tìm ra những bổ đề này là bài toán thú vị và không có một giải pháp chung. Nó phụ thuộc vào từng hệ thống và các tình huống cụ thể trong quá trình chứng minh hệ thống.

5. Kết luận

Bài báo này đã trình bày một phương pháp đặc tả và kiểm chứng các thuộc tính bất biến của các hệ đa tác tử nơi mà không gian trạng thái của chúng là vô hạn. Với bài toán này, các phương pháp kiểm chứng mô hình không thể giải quyết được. Trong phương pháp đề xuất, chúng tôi sử dụng ngôn ngữ đại số để đặc tả hành vi của các tác tử, hành vi của hệ thống và các thuộc tính bất biến cần chứng minh. Từ các đặc tả này, chúng ta xây dựng không gian trạng thái của hệ thống một cách đệ quy. Với cách tiếp cận này, chúng ta không cần chỉ ra tất cả các trạng thái của hệ thống như các phương pháp kiểm chứng mô hình. Không gian trạng thái của hệ thống gồm trạng thái khởi tạo và tập các quy luật để chuyển đến các trạng thái tiếp theo từ một trạng thái bất kỳ. Việc chứng minh tính đúng đắn của hệ thống đối với các thuộc tính bất biến được thực hiện bằng cách sử dụng tư tưởng quy nạp toán học. Trước tiên, chúng ta chứng minh thuộc tính đúng tại trạng thái khởi tạo. Giả sử thuộc tính đúng tại một trạng thái bất kỳ, ta chứng minh thuộc tính đúng tại tất cả các trạng thái tiếp theo. Bằng phương pháp này, chúng ta kết luận thuộc tính đúng tại mọi trạng thái của hệ thống.

Tuy nhiên, phương pháp đề xuất chỉ áp dụng chứng minh các thuộc tính bất biến. Với các hệ đa tác tử, có rất nhiều thuộc tính khác cần được chứng minh. Chúng tôi đang tập trung mở rộng phương pháp này cho cac thuộc tính khác. Hơn nữa, phương pháp đề xuất chỉ đúng khi đặc tả của hệ thống là đúng đắn. Trong khi đó, đặc tả này được xây dựng thủ công nên việc đảm bảo tính đúng đắn của nó là một vấn đề khó và chưa có lời giải thỏa đáng. Đây cũng là một trong những định hướng nghiên cứu quan trọng không những cho chứng minh tự động mà còn cho hầu hết các nghiên cứu về kiểm chứng mô hình và kiểm thử tự động [10]. Chúng tôi cũng sẽ áp dụng phương pháp đề xuất cho các hệ thống khác với độ phức tạp và quy mô lớn hơn để chỉ rõ tính hiệu quả của phương pháp đề xuất.

Lời cảm ơn

Nghiên cứu trong bài báo này được thực hiện dưới sự tài trợ của đề tài mã số CN.10.03 do Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội tài trợ.

Tài liệu tham khảo

[1] L. Alex, G. Hayzelden, Rachel A. Bourne, *Agent Technology for Communication Infrastructures*, JOHN WILEY & SONS Press, 2001

[2] K. Sycara, Multiagent Systems, *AI Magazine,* vol. 19, no. 2 (1998) 79.

[3] G. Weiss, *Multiagent Systems: a Modern Approach to Distributed Artificial Intelligence*, MIT Press, 1999

[4] M. Wooldridge, *An Introduction to MultiAgent Systems*, John Wiley & Sons, 2002

[5] M. Edmund, Jr. Clarke, Orna Grumberg, Doron A. Peled, *Model Checking,* MIT Press, 1999

[6] P. N. Hung, T. Aoki, T. Katayama, *Modular conformance testing and assume-guarantee verification for evolving component-based software*, IEICE Trans. Fundamentals, Special Issue on Theory of Concurrent Systems and Its Applications, vol. E92-A, no. 11 (2009) 2772.

[7] J. Magee, J. Kramer, *Concurrency: State Models & Java Programs*, John Wiley & Sons, 1999

[8] CafeOBJ official homepage, http://www.ldl.jaist.ac.jp/cafeobj/

[9] Kokichi FUTATSUGI, *Verifying Specifications with Proof Scores in CafeOBJ*, The 21st IEEE International Conference on Automated Software Engineering (ASE'06), 2006

[10] Paul C. Jorgensen, *Software Testing: A Craftman's Approach*, CRC Press, August 1995.

Specification and Verification of Invariant Properties of Multi-agent Systems

Pham Ngoc Hung[1], Dao Anh Hien[2], Nguyen Anh Nguyet[1], Nguyen Viet Ha[1]

[1]*VNU University of Engineering and Technology, 144 Xuan Thuy, Hanoi, Vietnam*
[2]*Hung Yen University of Technology and Education, Dan Tien, Khoai Chau, Hung Yen*

Theorem proving has been recognized as an important approach in improving the reliability of multi-agent systems. This paper proposes a method for specifying and proving invariant properties of multi-agent systems where their state spaces are infinite. In this method, the behaviors of multi-agent systems and their invariant properties are specified by using algebra specification. The state space of each multi-agent system is defined recursively by identifying the initial state and all possible next states of any state of the system. The proposed method then uses the induction proof in order to verify correctness of the system. A case study about the mutual exclusion problem for multi-agent system is also presented and discussed for showing the usefulness of the proposed method.