Tạp chí Khoa học Trường Đại học Cần Thơ
website: sj.ctu.edu.vn

ĐA TRỪU TƯỢNG KẾT HỢP TINH LỌC TRONG KIỂM TRA MÔ HÌNH HƯỚNG KÝ HIỆU

Bùi Hoài Thắng[1] và Nguyễn Thị Hồng Phấn[1]
[1] *Khoa Khoa học & Kinh tế Máy tính, Trường Đại học Bách Khoa Thành phố Hồ Chí Minh*

Thông tin chung:
Ngày nhận: 03/09/2013
Ngày chấp nhận: 21/10/2013

Title:
Multiple abstraction refinement in symbolic model checking

Từ khóa:
Tinh lọc, đa trừu tượng, kiểm tra mô hình hướng ký hiệu

Keywords:
Multiple abstraction refinement, symbolic model checking

ABSTRACT

In model checking, the state space explosion prevents verifying the large systems because exhaustive search fails to find errors. The problem becomes much more serious recently when the size and the complexity of the system-under-check are constantly growing up. Abstraction is the most effective method to overcome this problem. It is an approach used to reduce the size (of the state space) of the model in the reality. However, how to use abstraction effectively in reducing model checking efforts is still a question.

This work proposes a multiple abstraction refinement technique in symbolic model checking that allows to verify large state space systems. The proposed technique employs the abstraction refinement algorithm CEGAR proposed by Clarke et al. in 2000 combining with the multiple abstraction method of Qian and Nymeyer. Instead of checking on concrete model, an abstract model is verified first. If a counter-example is found, the searching scope will be widen on the next abstract model and so on, even on the original model. This process can be called refinement. The experimental results show that the new approach improves the performance of the model checking process.

TÓM TẮT

Trong kiểm tra mô hình, bùng nổ không gian trạng thái gây trở ngại đến việc kiểm chứng các hệ thống lớn vì quá trình tìm kiếm lỗi có thể không hoàn thành. Vấn đề càng trở nên nghiêm trọng hơn khi kích thước và sự phức tạp của hệ thống cần kiểm tra đang liên tục lớn dần lên. Trừu tượng là phương pháp hiệu quả nhất để giải quyết vấn đề này. Đây là một cách tiếp cận để giảm kích thước (của không gian trạng thái) của mô hình trong thực tế. Tuy nhiên, làm thế nào để sử dụng trừu tượng một cách hiệu quả trong việc giảm chi phí kiểm tra mô hình vẫn còn là câu hỏi.

Công trình này đề xuất kỹ thuật tinh lọc đa trừu tượng trong kiểm tra mô hình cho phép kiểm chứng các hệ thống có không gian trạng thái lớn. Phương pháp đề xuất là sự tận dụng giải thuật CEGAR được Clarke đề xuất vào năm 2000 kết hợp với phương pháp đa trừu tượng của Qian & Nymeyer. Thay vì kiểm tra trên mô hình thực, mô hình trừu tượng được kiểm chứng đầu tiên. Nếu tìm thấy counter-example, phạm vi tìm kiếm được mở rộng trên các mô hình trừu tượng tiếp theo và cứ thế, thậm chí là trên mô hình gốc. Quá trình này được gọi là tinh lọc. Kết quả thực nghiệm cho thấy rằng phương pháp mới cải thiện hiệu suất của quá trình kiểm tra mô hình.

1 GIỚI THIỆU

Trong công nghiệp, hầu hết các nhà sản xuất đều mong muốn sản phẩm của họ thâm nhập thị trường nhanh chóng với chất lượng cao. Với mục tiêu phải bảo đảm chất lượng, trong lĩnh vực công nghệ thông tin và truyền thông có nhiều phương pháp kiểm định chất lượng được áp dụng vào dây chuyền sản xuất như chứng minh đúng đắn (theorem proving), *kiểm tra mô hình* (model checking)[3] và kiểm thử (testing). Mỗi phương pháp có sức mạnh riêng và người kỹ sư hệ thống phải xem xét phương pháp nào thích hợp cho từng nhu cầu cụ thể. *Kiểm tra mô hình* được yêu thích vì sự tự động hóa và nó có thể được áp dụng vào kiểm tra cho các hệ thống phần cứng và phần mềm chưa hoàn chỉnh.

Trong kiểm tra mô hình, *mô hình* là sự đặc tả của hệ thống cần được kiểm tra, mô hình được sử dụng cho trong quá trình tìm kiếm các vi phạm các thuộc tính cần kiểm tra. Khi một vi phạm được tìm thấy, một counter-example được sinh ra để giúp các kỹ sư định vị được lỗi. Không may, khi kích thước và độ phức tạp của các hệ thống cần kiểm tra lớn liên tục, quá trình kiểm tra trở nên khó khăn hơn.

Gần đây, *kiểm tra mô hình hướng ký hiệu* trở thành hướng nghiên cứu được quan tâm và phổ biến nhất trong kiểm tra mô hình. Nó làm việc trên một tập các trạng thái và sự dịch chuyển giữa các trạng thái hơn là các yếu tố độc lập và biểu diễn các tập hợp là các biểu thức Boolean (của các biến). Vì vậy, các hệ thống có không gian trạng thái lớn có thể được kiểm chứng trên thực tế [8]. Thật ra, sự bùng nổ trạng thái vẫn xảy ra trong quá trình tìm kiếm khi kích thước của các không gian trạng thái lớn dần theo hàm mũ. Để giải quyết vấn đề bùng nổ, kỹ thuật trừu tượng có thể được dùng để giảm kích thước của mô hình.

Tuy nhiên, một phương pháp trừu tượng tốt vẫn đưa ra counter-example giả. Clarke *et al*. [4] đã đề xuất việc *tinh lọc* trên các counter-example giả có thể bị loại bỏ (abstraction refinement). Thay vì Clarke sử dụng phương pháp tinh lọc CEGAR, Qian K. *et al.* [13] giới thiệu kỹ thuật *đa trừu tượng* (multiple abstraction) để tinh lọc trừu tượng. Nếu mô hình trừu tượng không có lỗi, hệ thống được kiểm chứng là không có lỗi mà không cần kiểm tra mô hình gốc ban đầu. Dĩ nhiên, nếu tất cả các mô hình trừu tượng chứa lỗi, hệ thống thực sẽ được kiểm tra. Ngoài ra, tinh lọc trừu tượng phải giải quyết bùng nổ không gian trạng thái nếu phải thực hiện tinh lọc counter-example giả trên mô hình gốc. *Đa trừu tượng* cũng đã giải quyết cùng vấn đề này khi xem tất cả các mô hình trừu tượng và mô hình gốc độc lập mà không sử dụng lại counter-example từ các bước tìm kiếm trước đó để dẫn hướng tìm lỗi.

Trong bài báo cáo này, chúng tôi đề xuất một phương pháp tìm kiếm lỗi tận dụng ưu điểm của *tinh lọc trừu tượng* [4] và *đa trừu tượng* [3] được gọi là **MAR** (multiple abstraction refinement). Nói cách khác, giải thuật MAR là sự kết hợp của đa trừu tượng và tinh lọc trừu tượng. Hiện tại chỉ hỗ trợ kiểm tra mô hình trực tiếp (directed model checking).

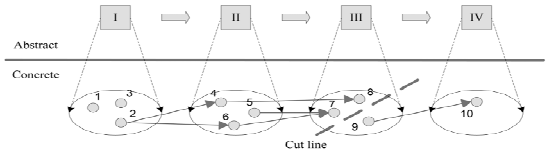
Phần còn lại của bài báo được tổ chức như sau: phần 2 trình bày về các công trình liên quan (giải thuật *CEGAR*, *đa trừu tượng* và phương pháp phân tích biến VRK). Các công trình này là tiền đề cung cấp kiến thức cho giải thuật mới được đề xuất *giải thuật đa trừu tượng kết hợp tinh lọc*, kết quả thực nghiệm cũng được trình bày trong phần này. Phần cuối trình bày kết luận và đề xuất.

2 PHƯƠNG PHÁP NGHIÊN CỨU

2.1 Công trình liên quan

2.1.1 Giải thuật CEGAR

Clarke *et al.* 2000 [4] đề xuất giải thuật tinh lọc trừu tượng dùng counter-example dẫn hướng (Counter-Example Guided Abstraction Refinement). Nếu một counter-example được tìm thấy trên mô hình trừu tượng, nó sẽ được xác nhận lại trên mô hình gốc. Đầu tiên, mô hình trừu tượng được sinh ra dựa trên mô hình gốc bằng hàm trừu tượng. Kế đó, giải thuật kiểm tra được vận dụng vào mô hình trừu tượng để tìm lỗi. Nếu không có lỗi trên mô hình trừu tượng, giải thuật dừng lại và hệ thống cần kiểm tra được trả lời là đúng với các thuộc tính mong muốn. Nếu một counter-example giả được tìm thấy, bộ kiểm chứng sẽ tinh lọc nó trên mô hình gốc. Kết quả cuối cùng sẽ được xác nhận đó có thật sự là counter-example hay không và báo cáo lên người dùng. Với giải thuật này, mức độ hiệu quả của công cụ kiểm tra phụ thuộc vào hàm trừu tượng và giải thuật tinh lọc. Phương pháp vẫn đối mặt với vấn đề như*: loop counter-example* [4], khi một counter-example trên mô hình trừu tượng có chứa vòng lặp, và *non-strong connectivity* (khi một trạng thái trừu tượng đại diện cho một tập các trạng thái thực không được kết nối với nhau). Trạng thái trừu tượng *III* trong Hình 1 là ví dụ của trường hợp thứ hai.



Hình 1: Hệ thống trừu tượng [7]

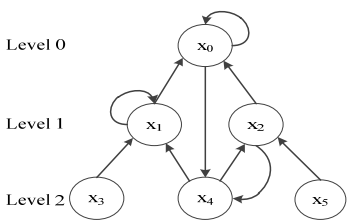
2.1.2 Giải thuật đa trừu tượng

Qian K. *et al.*[13] đề xuất chiến thuật *đa trừu tượng* để giảm chi phí kiểm tra. Dựa trên giả thuyết có các chuỗi mô hình trừu tượng $M_0 \subseteq M_1 \subseteq \ldots$ sao cho trạng thái trừu tượng S^i trên mô hình M_i được tách ra nhiều trạng thái S^{i1}, S^{i2},… trên M_{i+1} (mức độ trừu tượng của M_i nhiều hơn M_{i+1},). Giải thuật tìm kiếm chạy qua các mô hình trừu tượng theo thứ tựM_0, M_1,…tìm kiếm các vi phạm trong các mô hình trừu tượng này. Quá trình tìm kiếm có thể dừng lại nếu không tìm thấy bất kỳ counter-example nào trong bất cứ M_i. Ngược lại, quá trình tìm kiếm tiếp tục phát hiện ra lỗi trên các mô hình trừu tượng còn lại thậm chí là trên mô hình gốc. Giải thuật đa trừu tượng đối xử với tất cả các mô hình trừu tượng như các mô hình riêng rẽ và theo thứ tự. Nó chỉ xem thông tin có được về counter-example của mô hình trừu tượng trong lần tìm kiếm trước đó là heuristic. Dĩ nhiên, vấn đề các trạng thái trừu tượng có kết nối với nhau nhưng thật sự trên mô hình thật các trạng thái không có đường nối giữa chúng (non-strong connected abstract state) vẫn còn tồn tại khi kết quả có thể dẫn tới lần tìm kiếm kế tiếp bị sai đường. Trong trường hợp xấu nhất, giải thuật phải thực thi trên mô hình thật để tìm ra counter-example thật sự.

Tuy hiệu suất của giải thuật đa trừu tượng [13] rất tốt, nó vẫn cần được cải thiện để sử dụng lại thông tin từ giai đoạn tìm kiếm trước đó để chỉ tập trung vào các vùng đáng nghi ngờ trong mô hình trừu tượng kế tiếp có khả năng chứa các vi phạm.

2.1.3 Phương pháp phân tích biến VRK

Phần này giới thiệu sơ lược về phương pháp phân tích biến VRK [19]. Đây là một phương pháp để tạo ra mô hình trừu tượng. Bài báo [19] áp dụng ý tưởng giải thuật PageRank vào phân tích biến để tính độ quan trọng của biến trong đồ thị phụ thuộc biến (xem Hình 2).



Hình 2: Đồ thị phụ thuộc biến[18]

Giải thuật phân tích biến hội tụ VRK được mô tả như sau:

a. Áp dụng giải thuật PageRank để tính toán độ quan trọng của mỗi biến trong đồ thị:

$$PR(A) = \frac{(1-d)}{N} + d\left(\sum_{B \in Parent(A)} \frac{PR(B)}{N(B)}\right)$$

N là số lượng biến, *Parent(A)* là một tập tất cả các biến mà A phụ thuộc vào, *N(B)* là số lượng các biến bị phụ thuộc vào B,d là hệ số damping được thiết lập giá trị giữa 0 và 1. Theo Brin *et al.* [1], giá trị tốt nhất cho d là 0.85.

b. Điều chỉnh mức độ quan trọng của biến có được từ bước i

Dựa vào mức (level) của biến trên đồ thị, thực hiện điều chỉnh lại độ quan trong của biến theo công thức sau:

$$E(x) = E(x) + \frac{MaxLevel - Level(x)}{MaxLevel}$$

MaxLevel là giá trị lớn nhất của level có trong đồ thị, *Level(x)* là level của biến x trên đồ thị.

2.2 Giải thuật đề xuất

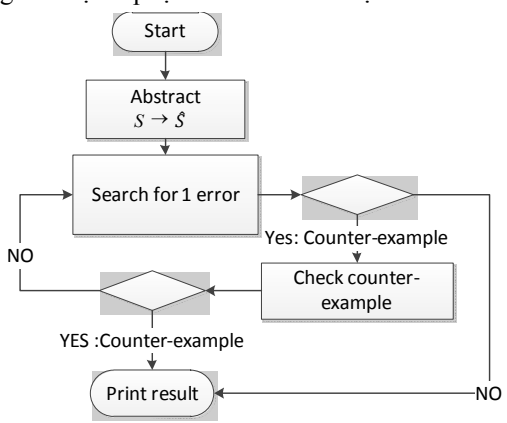
Dựa trên giải thuật *đa trừu tượng* và *CEGAR*, chúng tôi đề xuất một phương pháp mới với ưu điểm của cả hai. Với *CEGAR*, bất cứ khi nào một counter-example được tìm thấy trên các mô hình trừu tượng, nó sẽ được tinh lọc trên hệ thống gốc. Với *đa trừu tượng*, việc tìm kiếm sẽ được thực hiện trên mô hình trừu tượng kế tiếp (theo mức độ trừu tượng của các mô hình giảm dần) mỗi khi lỗi được tìm thấy trên các mô hình trừu tượng trước, quá trình tìm kiếm không sử dụng lại thông tin về lỗi ở mô hình trước, đây cũng chính là nhược điểm của giải thuật. Giải thuật mới vận dụng ý tưởng *đa trừu tượng* từ [13] và *tinh lọc* counter-example từ CEGAR [4].

Giống như *đa trừu tượng*, giải thuật mới sẽ duyệt lần lượt qua một chuỗi các mô hình trừu tượng $M_0 \subseteq M_1 \subseteq \ldots$, để tìm kiếm các vi phạm (mức độ trừu tượng M_0 là nhiều nhất). Counter-example tìm thấy trên M_0 sẽ được xác nhận lại trên M_1 và tiếp tục theo thứ tự mức độ trừu tượng M_2, M_3...và sau đó là trên hệ thống gốc. Counter-example tìm thấy trên hệ thống gốc là một chứng cứ cho việc vi phạm các thuộc tính cần kiểm tra.

Một chuỗi quá trình kiểm tra có thể được kết thúc sớm hơn nếu counter-example không được xác nhận lại trên bất cứ mô hình trừu tượng M_i nào (kể cả mô hình gốc). Trong trường hợp này, lỗi được tìm thấy trên M_0 sẽ bị từ chối (vì M_i xác nhận

không phải lỗi) và quá trình tìm kiếm trên M_0 sẽ tiếp tục.

Giải thuật được minh họa như trong Hình 3. Trong sơ đồ này, khối *Checkcounter-example* là hàm đệ qui gọi chính nó nhiều lần với mô hình trừu tượng M_i hoặc mô hình gốc lúc cuối. Nếu không có counter-example được tìm thấy trên M_0, quá trình kiểm tra kết thúc, báo cáo kết quả không có lỗi. Nếu quá trình xác nhận counter-example thất bại, giải thuật tiếp tục tìm kiếm cho một lỗi khác.



Hình 3: Quá trình tinh lọc trừu tượng

2.1 Thực nghiệm

2.1.1 Môi trường thực nghiệm

Trong thử nghiệm này, chúng tôi sử dụng lại các mô hình được cung cấp từ trang web NuSMV. Các mô hình này là các mô hình đúng được sử dụng để kiểm chứng giải thuật tìm lỗi mới. Các mô hình được sử dụng là ***brp***, ***dme***, ***gigamax***, ***msi_wtra***, ***periodic***.

Các thử nghiệm được thực hiện trên máy tính core i3-2330M 2.2 GHz với 4MB RAM chạy hệ điều hành Ubuntu 12.04 với công cụ Golfer[1]. Kết quả được được tính trung bình của các lần chạy theo đơn vị thời gian là giây.

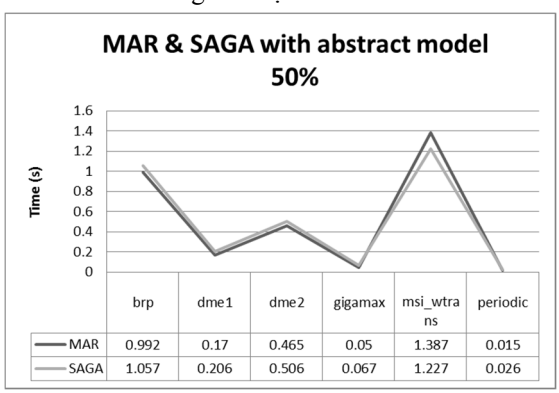
Trong thử nghiệm này, chúng tôi so sánh thời gian tìm lỗi của giải thuật mới đề xuất với giải thuật đa trừu tượng và giải thuật tìm lỗi gốc của NuSMV (giải thuật vét cạn, BFS trong bài bài toán invariant model checking). Rất tiếc là chúng tôi không thể so sánh với giải thuật CEGAR vì chúng tôi không có công cụ hiện thực của Clarke.

[1] Golfer là công trình của (Qian & Nymeyer, 2004-2005) và (Bui & Nymeyer, 2008-2009) dựa trên NuSMV.

2.1.2 Kết quả thực nghiệm

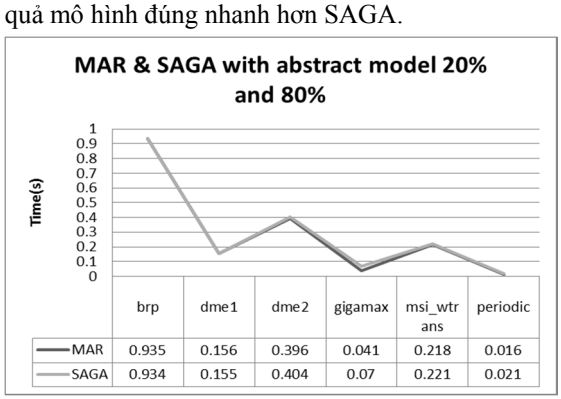
Trong phần này, chúng tôi so sánh giữa giải thuật *đa trừu tượng kết hợp tinh lọc* (gọi tắt là **MAR**) và giải thuật *đa trừu tượng* (gọi là **SAGA**) khi áp dụng cùng một phương pháp phân tích biến VRK (tham khảo[19]).

Với một mô hình trừu tượng (giữ lại 50% biến) ở Hình 4, trường hợp này giải thuật MAR chỉ dùng một mô hình trừu tượng, nên có thể xem nó tương tự như giải thuật CEGAR. Nhìn chung, giải thuật MAR nhanh hơn giải thuật SAGA.



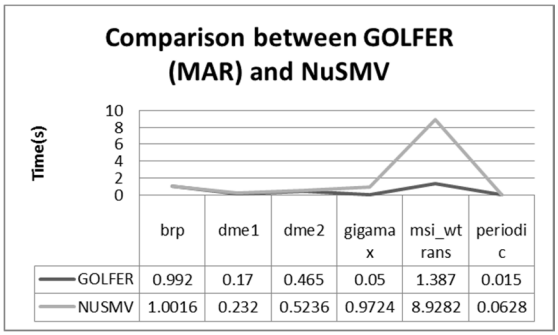
Hình 4: So sánh MAR và SAGA với chỉ một mô hình trừu tượng

Với những ưu điểm của giải thuật *đa trừu tượng kết hợp tinh lọc*, chúng tôi có thể đặc tả một chuỗi danh sách các mô hình trừu tượng. Công cụ kiểm tra có thể ngừng lại giữa quá trình kiểm chứng mà không cần kiểm tra hết tất cả các mô hình trừu tượng và mô hình gốc. Trong thực nghiệm này, chúng tôi sử dụng hai mô hình trừu tượng được loại bỏ 80% và 20% biến dựa trên giải thuật VRK (tương ứng giữ lại 20% và 80% biến theo độ quan trọng). Kết quả cho thấy như ở Hình 5. Một lần nữa, nhìn tổng quát MAR báo cáo kết quả mô hình đúng nhanh hơn SAGA.



Hình 5: So sánh MAR và SAGA với hai mô hình trừu tượng

Để chắc chắn tính hữu dụng của giải thuật mới, chúng tôi cũng thực hiện so sánh nó với giải thuật NuSMV ban đầu. Kết quả ở Hình 6 chỉ ra rằng giải thuật MAR là tốt nhất.



Hình 6: So sánh MAR với giải thuật tìm kiếm NuSMV

3 KẾT LUẬN VÀ ĐỀ XUẤT

3.1 Kết luận

Công trình này giới thiệu một kỹ thuật kiểm tra mô hình dựa trên việc *đa trừu tượng kết hợp tinh lọc* counter-example được đề xuất. Kỹ thuật mới có được ưu điểm của hai kỹ thuật trước đó. Kỹ thuật mới sử dụng *đa trừu tượng* để xác nhận lại counter-example được tìm thấy trên mô hình trừu tượng nhiều nhất (mô hình trừu tượng được lược bỏ hầu hết các biến) và sử dụng ý tưởng *tinh lọc* counter-example để tinh lọc counter-example (giải thuật này cho phép tinh lọc trên nhiều mô hình trừu tượng trước khi kiểm tra trên mô hình gốc so với CEGAR chỉ sử dụng duy nhất một mô hình trừu tượng và mô hình gốc). Sự cải tiến ở đây cho phép kiểm tra trên nhiều mô hình trừu tượng có kích thước không gian trạng thái nhỏ hơn mô hình gốc và chính vì thế mà có thể giảm công sức kiểm tra.

Kết quả thử nghiệm đã khẳng định các giả thuyết. Dĩ nhiên, giải thuật của chúng tôi vẫn phải đối mặt cùng vấn đề của *CEGAR* hoặc giải thuật *đa trừu tượng* nếu mô hình gốc có chứa lỗi. Trong trường hợp đó, giải thuật sẽ vẫn phải tìm kiếm lỗi trên mô hình gốc để sinh ra counter-example.

3.2 Đề xuất

Giải thuật cần được mở rộng để hỗ trợ các hướng kiểm tra khác (thay vì hiện nay chỉ hỗ trợ kiểm tra bất biến – invariant model checking). Đây sẽ là công việc của tương lai.

LỜI CẢM TẠ

Nhóm tác giả xin chân thành cảm ơn sự hỗ trợ kinh phí của Trường Đại học Bách Khoa Thành phố Hồ Chí Minh thông qua đề tài C2013-20-07/HĐ-KHCN.

TÀI LIỆU THAM KHẢO

1. Brin, Sergey and Page, Lawrence. The anatomy of a largescale hyper-textual Web search engine. *InProc. of WWW7*, pages 107-117, Brisbane, Australia, 1998.
2. Clarke E. M., etal. Formal methods: state of the art and future directions. ACM Computing Surveys, 1996.
3. Clarke E. M., Grumberg O., and Peled D. A., *Model Checking*, MIT Press, 1999
4. Clarke E. M., Grumberg O., Jha S., Lu Y., and Veith H., Counter-example-Guided Abstraction Refinement*, CAV'00, LNCS*, vol. 1855, pp. 154-169, 2000.
5. Dennis Dams: Abstraction in Software Model Checking: Principles and Practice (Tutorial Overview and Bibliography). *SPIN 2002*: 14-21.
6. Edelkamp S., Lluch-Lafuente A., and Leue S., Directed Explicit Model Checking with HSF-SPIN, *SPIN'01, LNCS*, vol. 2057, pp. 57-79, 2001.
7. Fei He, Xiaoyu Song, William N.N. Hung, Ming Gu, Jiaguang Sun, "Integrating Evolutionary Computation with Abstraction Refinement for Model Checking," *IEEE Transactions on Computers*, vol. 59, no. 1, pp. 116-126, Jan. 2010, doi:10.1109/TC.2009.105.
8. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. 1992. Symbolic model checking: 1020 states and beyond. Inf. Comput. 98, 2 (June 1992), 142-170.
9. Menzies T., Owen, D., and Richardson J., The Strangest Thing about Software, *IEEE Computer*, vol. 40, no. 1, pp. 54-60, 2007.
10. Pelanek R., Typical structural properties of state spaces, *LNCS*, vol. 2989, pp. 5-22, 2004.
11. Qian K., Nymeyer A., Guided Invariant Model Checking Based on Abstraction and Symbolic Pattern Databases, *TACAS'04, LNCS*, vol. 2988, pp. 497-511, 2004.
12. Qian K., Nymeyer A., Abstraction-guided model checking using symbolic IDA* and heuristic synthesis, *FORTE'05, LNCS*, vol. 3731, pp. 275-289, 2005.
13. Qian K., Nymeyer A, and Susanto S.. Experiments with Multiple Abstraction

Heuristics in Symbolic Verification, *SARA 2005, LNAI 3607*, pp. 290-304, 2005.

14. Qing Cui, Alex Dekhtyar., On Improving Local Website Search Using Web Server Traffic Logs: *A Preliminary Report*, 2005.
15. Thang. H. Bui, Nymeyer A., Formal Verification Based on Guided Random Walks, *iFM'09, LNCS*, vol. 5423, pp. 72-87, 2009a.
16. Thang. H. Bui, Nymeyer A., Formal Model Simulation: Can it be Guided?, *SSBSE'09*, pp. 93-96, IEEE CS, 2009b.
17. Thang. H. Bui, Nymeyer A., Heuristic Sensitivity in Guided Random-Walk Based Model Checking, *SEFM'09*, pp. 125-134, IEEE CS, 2009c.
18. Thang. H. Bui, Phuong. B.K. Dang, Yet another variable dependency analysis for abstraction guided model checking, *SEATUC 2012*.
19. Thang. H. Bui, Phan. T. H. Nguyen, Convergent Propagtion variable dependency analysis for multiple abstraction model checking, *SEATUC 2013.*
20. Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar and Gregoire Sutre.Lazy Abstraction. *In ACM SIGPLAN-SIGACT Conference on Principles of Programming Languages*, pages 58-70, 2002.