Nghiên cứu phương pháp sinh dữ liệu kiểm thử phần mềm dựa trên kỹ thuật kiểm chứng mô hình

Phan Văn Tiến

Trường Đại học Công nghệ
Luận văn ThS. ngành: Công nghệ phần mềm; Mã số: 60 48 10
Người hướng dẫn: TS. Nguyễn Trường Thắng
Năm bảo vệ: 2011

Abstract. Trình bày cơ sở lý luận về kiểm định phần mềm và các nhóm kiểm định phần mềm. Giới thiệu về JPF, kiến trúc của JPF, cách mở rộng, phát triển trên JPF. Ngoài ra giới thiệu về thực thi tượng trưng để sinh dữ liệu kiểm thử cho chương trình trong JPF cũng như cho phép sinh tự động dữ liệu kiểm thử chương trình Java. Tìm hiểu về SMT, Z3, các lý thuyết được hỗ trợ trên Z3, các API của Z3 để tích hợp với JPF và ứng dụng của Z3. Nghiên cứu, đánh giá các giải pháp như: Kiến trúc hệ thống, chuyển đổi dữ liệu, thiết kế và cài đặt.

Keywords. Công nghệ phần mềm; Dữ liệu; Kiểm chứng mô hình; Phần mềm

Content

Trong những năm gần đây, việc phát triển phần mềm ngày càng được chuyên nghiệp hóa. Các phần mềm được phát triển ngày càng có quy mô lớn. Yêu cầu đảm bảo chất lượng phần mềm là một trong những mục tiêu quan trong nhất, đặc biệt trong một số lĩnh vực như y khoa, ngân hàng, hàng không… Việc kiểm thử, kiểm chứng phần mềm một cách thủ công chỉ đảm bảo được phần nào chất lượng của phần mềm. Vì vậy rất nhiều các tổ chức, công ty đã nghiên cứu và phát triển các lý thuyết cũng như công cụ để kiểm chứng, kiểm thử phần mềm một cách tự động.

Xuất phát từ nhu cầu thực tế trên, tác giả đã nghiên cứu một số lý thuyết, công cụ trong việc kiểm chứng và kiểm thử phần mềm. Một lý thuyết nền tảng rất quan trọng đó là lý thuyết về tính thỏa được, viết tắt là SMT (Satisfiability Modulo Theories). Lý thuyết về tính thỏa được đã được ứng dụng để giải quyết nhiều bài toán trong công nghệ phần mềm như:

- Kiểm chứng chương trình
- Khám phá chương trình
- Mô hình hóa phần mềm
- Sinh các ca kiểm thử

Hiện nay Microsoft Z3 là một công cụ tìm lời giải cho SMT đang được áp dụng trong nhiều dự án của Microsoft như: Pex, Spec#, SLAM/SDV, Yogi. Z3 được đánh giá là công cụ tìm lời giải mạnh nhất hiện nay. Tuy nhiên Z3 chỉ được áp dụng cho các ngôn ngữ của Microsoft. Vì vậy tác giả đặt ra vấn đề: Liệu có thể sử dụng Z3 để kiểm chứng cho các chương trình viết bằng ngôn ngữ khác như Java?

Trong quá trình nghiên cứu về kiểm chứng chương trình tác giả cũng có tìm hiểu về JavaPathFinder (JPF). JPF là một dự án mã nguồn mở được phát triển trên ngôn ngữ Java. Hiện nay có một mở rộng của JPF trong việc sinh tự động dữ liệu đầu vào để kiểm thử chương trình. Tuy nhiên còn rất nhiều hạn chế, vì vậy tác giả đã nghĩ đến việc làm sao để tích hợp được Z3 với JPF để có thể sinh tự động dữ liệu kiểm thử chương trình. Nếu việc tích hợp thành công thì sẽ dẫn tới việc giải quyết được lớp bài toán rộng hơn. Điều này là rất có ý nghĩa đối với thực tế.

Mục tiêu đề tài:

Mục tiêu của đề tài là nghiên cứu nắm bắt rõ về Z3 và JPF. Sau đó bước đầu tích hợp thành công Z3 và JPF để có thể sinh tự động dữ liệu kiểm thử chương trình Java cho các bài toán mà hiện nay JPF không thể thực hiện được. (ví dụ: sinh tự động dữ liệu cho số học phi tuyến tính).

CẤU TRÚC CỦA LUẬN VĂN

Luận văn bao gồm các phần sau:

Mở đầu: Giới thiệu về đề tài, tính cấp thiết cũng như mục tiêu của đề tài

Chương 1: Cơ sở lý luận

Chương 2: JPF và Thực thi tượng trưng

Nội dung: Giới thiêu JPF là gì? Kiến trúc của JPF, cách mở rộng, phát triển trên JPF. Ngoài ra còn một phần rất quan trọng đó là giới thiệu về thực thi tượng trưng để sinh dữ liệu kiểm thử cho chương trình trong JPF. Mở rộng này sẽ cho phép sinh tự động dữ liệu kiểm thử chương trình Java.

Chương 3: Microsoft Z3

Nội dung: Giới thiệu về lý thuyết tính thỏa được SMT, Z3, các lý thuyết được hỗ trợ trên Z3, các API của Z3 để tích hợp với JPF, các ứng dụng của Z3.

Chương 4: Tích hợp JPF với Z3

Nội dung: Nghiên cứu, đánh giá các giải pháp. Sau khi đã có giải pháp tiến hành thiết kế kiến trúc hệ thống, sau đó chi tiết hóa sang mức gói, mức lớp cuối cùng là cài đặt và đánh giá kết quả.

Kết luận và hướng phát triển của luận văn

Trình bày kết quả sau khi nghiên cứu, triển khai và hướng phát triển tiếp theo.

References

1. Clart Barret, Aaron Stump, Ceasar Timeli (2010), *The SMT-Lib Standard*, version 2.0, www.SMT-LIB.org
2. Java Path Finder, http://javapathfinder.sourceforge.net/
3. D. Detlefs, G. Nelson, and J. B. Saxe (2005), *Simplify: a theorem prover for program checking*, J. ACM, pp. 365-473
4. King, J.C (1976), *Symbolic Execution and testing*, communications of the ACM, pp. 385 - 394
5. Leonardo de Moura and Nikolaj Bjørner (2006), *Z3 – a tutorial*, Microsoft Research, USA
6. Leonardo de Moura and Nikolaj Bjørner (2008), *Z3: An Efficient SMT Solver*, Microsoft Research, One Microsoft Way, Redmond, vWA, 98074, USA, pp. 2-3