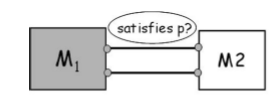
Đảm bảo giả định

Kiểm chứng đảm bảo giả định dựa trên sức mạnh của kĩ thuật chia để trị. Phân rã bài toán lớn là kiểm chứng một hệ thống lớn thành việc kiểm chứng các thành phần nhỏ hơn và độc lập với nhau. Vì thế, trong khi kiểm chứng, thay vì phải kiểm chứng một hệ thống lớn với số lượng trạng thái và hành vi rất lớn thì chúng ta chỉ việc kiểm chứng trên các thành phần nhỏ hơn và độc lập với nhau này. Điểm mấu chốt của quy tắc đảm bảo giả định là cân nhắc để giữa các thành phần không có sự cô lập với nhau nhưng lại có sự phối hợp với giả định về ngữ cảnh của giả định. Xét trường hợp đơn giản nhất hệ thống lớn ban đầu là một hệ thống M gồm 2 thành phần M1 và M2 (M1 || M2 = M).



Hình 1: Cái nhìn chung về kiểm chứng đảm bảo giả định

Bài toán đưa ra là kiểm chứng hệ thống M có thỏa mãn thuộc tính p nào đó hay không mà không cần ghép nối M1 với M2. Để làm được điều này, một ngữ cảnh giả định A(p) được sinh ra bằng cách áp dụng một số thuật toán học sao cho:

<A(p)> M1 <p> (1)

<true> M2 <A(p)> (2)

Trong mô hình đảm giả định, xét một công thức tổng quát <A(p)> M <p> (\*) với là một thành phần nào đó của hệ thống, p là một thuộc tính và A(p) là một giả định về môi trường của M. (\*) đúng nếu mỗi khi M­ là một phần của hệ thống thỏa mãn A(p) thì hệ thống thỏa mãn p. Để kiểm tra tính đúng đắn của (\*) với A(p) và p là các LTS an toàn, chúng ta có thể sử dụng công cụ LTSA [1] để ghép nối A(p) || M || perr và kiểm tra xem có trạng thái nào của hệ thống sau khi ghép nối đến được trạng thái π hay không. Nếu tồn tại trạng thái nào đó đến được trạng thái π, (\*) không thỏa mãn, ngược lại (\*) không thỏa mãn. Vấn đề còn lại là làm thế nào để sinh được ngữ cảnh giả định A(p) để thỏa mãn đồng thời cả (1) và (2). Trong giới hạn của luận văn này em xin trình bày hai phương pháp:

- Phương pháp sinh ngữ cảnh tường minh: Sử dụng thuật toán học L\*

- Phương pháp sinh ngữ cảnh không tường minh: Sử dụng thuật toán CDNF

[1] P. N. Hung, T. Aoki and T. Katayama: “Modular Conformance Testing and AssumeGuarantee Verification for Evolving Component-Based Software”, Conditional accepted by IEICE Trans. Fundamentals, Special Issue on Theory of Concurrent Systems and Its Applications (2009).