HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÀI GIẢNG MÔN HỌC

AN TOÀN HỆ ĐIỀU HÀNH

CHƯƠNG 5 – ĐÁNH GIÁ AN TOÀN HĐH

Giảng viên: PGS.TS. Hoàng Xuân Dậu

E-mail: dauhx@ptit.edu.vn

Khoa: An toàn thông tin

NỘI DUNG CHƯƠNG 5

1. Khái quát về đặc tả an toàn
2. Các kỹ thuật kiểm chứng đặc tả an toàn
3. Các phương pháp phân rã dữ liệu và thuật toán
4. Các kỹ thuật kiểm chứng mã chương trình

5.1 Khái quát về đặc tả an toàn

- Vấn đề đánh giá an toàn có liên quan chặt chẽ đến việc đảm bảo chất lượng phần mềm theo nghĩa phần mềm được xây dựng xong phải thỏa mãn các đặc tả về an toàn của phần mềm.
- Như vậy, các khiếm khuyết hoặc lỗi trong quá trình phát triển và triển khai phần mềm sẽ dẫn đến các lỗ hổng và có thể bị khai thác một cách vô tình hay có chủ đích.
- Bản thân hệ điều hành là tập hợp các phần mềm mà mỗi phần mềm cung cấp một số các chức năng của HĐH.
  - Như vậy an toàn của hệ điều hành lệ thuộc vào mức độ an toàn của các phần mềm chức năng.

Khái quát về đặc tả an toàn

- Đặc tả yêu cầu phần mềm (Software requirements specification) là một mô tả của phần mềm được phát triển.
  - Nó được mô phỏng theo đặc tả yêu cầu nghiệp vụ, còn được gọi là đặc tả yêu cầu của các bên liên quan.
  - Đặc tả yêu cầu phần mềm đưa ra các yêu cầu chức năng và phi chức năng;
  - Ngoài ra, nó có thể bao gồm một tập hợp các trường hợp sử dụng mô tả các tương tác của người dùng mà phần mềm phải cung cấp cho người dùng hỗ trợ tương tác hiệu quả.
- Đặc tả về an toàn của phần mềm là các mô tả về các yêu cầu an ninh, an toàn của phần mềm.

Khái quát về đặc tả an toàn

- Đặc tả an toàn vs. Mô hình an toàn:
  - Các đặc tả an toàn chỉ hữu ích cho hệ thống cần phải đảm bảo mức độ an ninh cao nhất;
  - Mô hình an toàn có khả năng ứng dụng rộng rãi hơn. Sử dụng mô hình an toàn giúp cho việc xây dựng các đặc tả an toàn được thuận tiện và dễ dàng hơn, tuy nhiên điều ngược lại không chính xác.
- Mục đích của đặc tả an toàn là diễn tả các hành vi chức năng của hệ thống theo cách thức chính xác, không mơ hồ và phù hợp với việc xử lý của máy tính.
  - Yêu cầu phù hợp với xử lý máy tính là để giảm thiểu lỗi do con người gây ra và giúp cho việc phân tích hệ thống được xây dựng có thỏa mãn các đặc tả hay không.

Khái quát về đặc tả an toàn

- Đặc tả an toàn chính tắc (formal security specification):
  - Các đặc tả chính tắc có thể dùng để chứng minh các thuộc tính về thiết kế của hệ thống, đặc biệt là việc hệ thống xây dựng phù hợp với các đặc tả của mô hình an toàn.
  - Công việc kiểm chứng chứng minh việc triển khai tuân thủ hay phù hợp với các đặc tả chính tắc.
  - Việc chứng minh một cách chính tắc và đầy đủ cho hệ thống lớn thực sự là một thách thức cho dù về mặt lý thuyết chứng minh đã được nghiên cứu rõ ràng.
  - Dù vậy, việc thực hiện kiểm chứng chính tắc một phần giúp người dùng đánh giá mức độ tương ứng giữa đặc tả và triển khai.

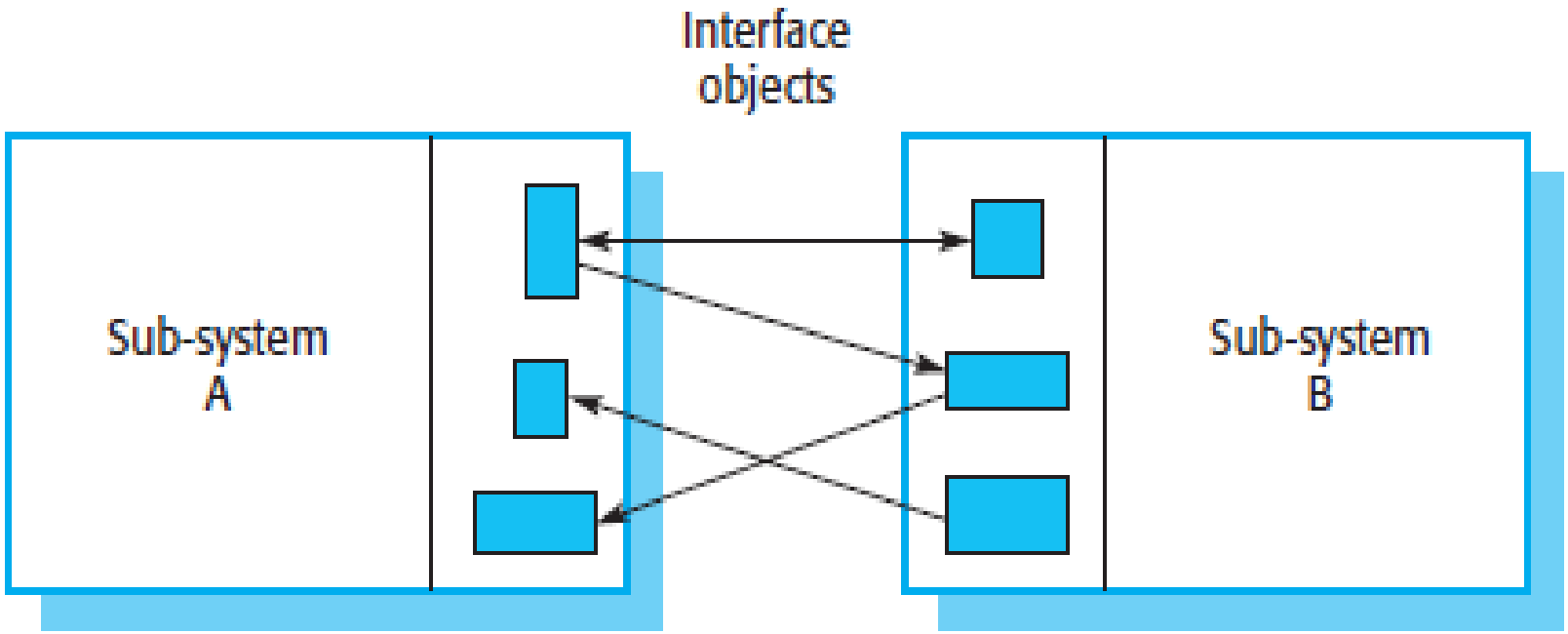
Khái quát về đặc tả an toàn

- Các đặc tả chính tắc trông giống như chương trình máy tính thông thường với các biểu thức lô-gíc và phép toán.
- Tuy nhiên, trong đặc tả chính tắc, các ký hiệu có ngữ nghĩa phong phú hơn ngôn ngữ máy tính cho phép biểu diễn các phép toán lô-gíc và quan hệ giữa chúng.

Khái quát về đặc tả an toàn

- Các đặc tả chính tắc bao gồm:
  - Đặc tả giao tiếp: giúp cho việc phân rã các hệ thống lớn thành các hệ thống con. Các giao tiếp thường được mô tả bằng tập các đối tượng hay thành phần cho biết dữ liệu và các thao tác được truy cập thông qua giao tiếp;
  - Đặc tả hành vi: mô tả các trạng thái có thể của hệ thống và các thao tác làm thay đổi trạng thái. Nói cách khác các hành vi của hệ thống có thể được biểu diễn bằng cách xây dựng cách thức các hành vi này làm thay đổi trạng thái của hệ thống như thế nào.

Mô phỏng giao tiếp giữa 2 hệ thống



Tương quan giữa mô hình và đặc tả an toàn

MODEL

Invariant: The system is secure if and only if, for all $s \in S$, $o \in O$,
if **r** $\in A(s,o)$ then $sclass(s) \geq oclass(o)$,
if **w** $\in A(s,o)$ then $oclass(o) \geq sclass(s)$.

Constraint 1: For all $o \in O$,
$'oclass(o) > oclass(o)$.

Constraint 2: For all $o \in O$,
if **r** $\notin A(subj,o)$
then for all $a \in S$, $'A(s,o) = A(s,o)$.

SPECIFICATION

```
  INVARIANT
    FOR_ALL (p:process,f:file) SUCH_THAT (file_exists(f) AND
proc_exists(p))
          (IF "r" IN access (p,f)
             THEN proc_class (p) >= file-class (f))
       & (IF "w" IN access (p,f)
             THEN file_class (f) >= proc_class (p))
  CONSTRAINTS
    FOR_ALL f:file SUCH_THAT file_exists (f)
         'file_class (f) >= file_class (f)
    FOR_ALL f:file SUCH_THAT file_exists (f)
         IF NOT ("r" IN access (cur_proc, f))
         THEN FOR_ALL p:process SUCH_THAT proc_exists (p)
                   'access (p, f) = access (p, f)
```

5.2 Các kỹ thuật kiểm chứng đặc tả an toàn

- Kiểm chứng đặc tả
- Kiểm chứng bằng Alloy (tự học)
  - Giới thiệu
  - Lập mô hình
  - Mô hình hệ thống file.

Kiểm chứng đặc tả

- Kiểm chứng hay chứng minh tính đúng của các đặc tả thường được thực hiện sử dụng các công cụ tự động do việc thực hiện thủ công hay gặp lỗi;
- Các phương pháp tiếp cận cho chứng minh đặc tả:
  - Chứng minh định lý (theorem prover) và
  - Kiểm chứng mô hình (model checker).

Chứng minh định lý

- Các công cụ này có thể có các mức độ tinh vi và phức tạp khác nhau từ việc kiểm tra các chứng minh các bước thủ công cho đến sử dụng các kỹ thuật trí tuệ nhân tạo.
  - Các hệ thống chứng minh và tích hợp đặc tả cho phép tạo ra một cách tự động các định lý dựa trên các tiên đề, hàm, bất biến, các hạn chế và các thành phần khác của đặc tả.
- Các công cụ hiện thời cho phép chứng minh các bất biến và các ràng buộc của các đặc tả chứa hàng nghìn dòng với mức độ tin cậy thỏa đáng.
  - Thông thường, các công cụ này cần có sự trợ giúp từ phía người dùng trong việc sinh ra các tiên đề, ràng buộc hay bất biến.

Kiểm chứng mô hình

- Kỹ thuật kiểm chứng mô hình dựa trên việc mô tả các hành vi có thể của hệ thống theo cách thức chính xác và rõ ràng về mặt toán học.
- Tiếp theo đó, các mô hình hệ thống được kiểm nghiệm tất cả các trạng thái có thể mà thỏa mãn các mô tả ở trên bằng thuật toán.
- Việc này cho phép phát hiện sớm các lỗi như thiếu đầy đủ, mơ hồ, không nhất quán trong giai đoạn phân tích thiết kế.

Kiểm chứng mô hình

- Kiểm chứng mô hình bao gồm cơ sở từ kiểm nghiệm toàn bộ mô hình hay kiểm nghiệm các tình huống giới hạn (mô phỏng) hay kiểm nghiệm thực tế (test).
- Đáng chú ý nhất là kiểm chứng toàn bộ mô hình:
  - Đây là kỹ thuật xem xét tất cả các trạng thái hệ thống có thể theo kiểu vét cạn.
  - Nhờ vậy có thể chứng minh được mô hình hệ thống cho trước thực sự thỏa mãn một thuộc tính nhất định được mô tả trong phần đặc tả.

Các công cụ kiểm chứng mô hình

- Spin :
  - Được dùng để lập mô hình phần mềm song song hay tiến trình dị bộ.
  - Được phát triển bởi Bell Labs, Spin chủ yếu nhắm đến kiểm chứng chính tắc các thuật toán máy tính.
- Uppaal :
  - Được dùng để lập mô hình hệ thống theo thời gian thực.
  - Các chức năng cũng tương tự như Spin.
  - Uppaal sử dụng ngôn ngữ riêng cho việc mô tả các mô hình cũng như các thuộc tính.

Các công cụ kiểm chứng mô hình

- SMV, NuSMV : Được dùng để lập mô hình phần cứng (lô-gíc số) tuy nhiên cũng có thể dùng được cho lĩnh vực khác.
- FDR : Được dùng để lập mô hình hệ thống dị bộ được phát triển bởi Trường Oxford.
  - FDR sử dụng ngôn ngữ mô tả dành cho các tiến trình song song.
  - Các hệ thống được lập mô hình dựa trên các sự kiện đồng bộ.
  - FDR có nhiều ảnh hưởng lên các công cụ khác như SPIN.

Các công cụ kiểm chứng mô hình

- Alloy : được dùng để phân tích tính nhất quán của các cấu trúc dữ liệu dựa trên lý thuyết tập hợp do Trường MIT phát triển.
  - Alloy sử dụng lô-gíc bậc nhất để chuyển các đặc tả thành các biểu thức Boolean và phân tích dựa trên bộ phân tích SAT.
- Simulink Design Verifier : được dùng để kiểm chứng mô hình được sinh ra từ Simulink, một công cụ mô phỏng dựa trên luồng dữ liệu và máy trạng thái.
  - Công cụ này được các kỹ sư sử dụng rộng rãi.
  - Điểm khác biệt là Simulink cho phép sinh ra mã C từ các mô hình.
  - Vì vậy, công cụ này cũng có thể coi là công cụ triển khai.

5.3 Các phương pháp phân rã dữ liệu và thuật toán

- Giới thiệu
- Phân rã cấu trúc dữ liệu
- Phân rã thuật toán

Phân rã dữ liệu và thuật toán - Giới thiệu

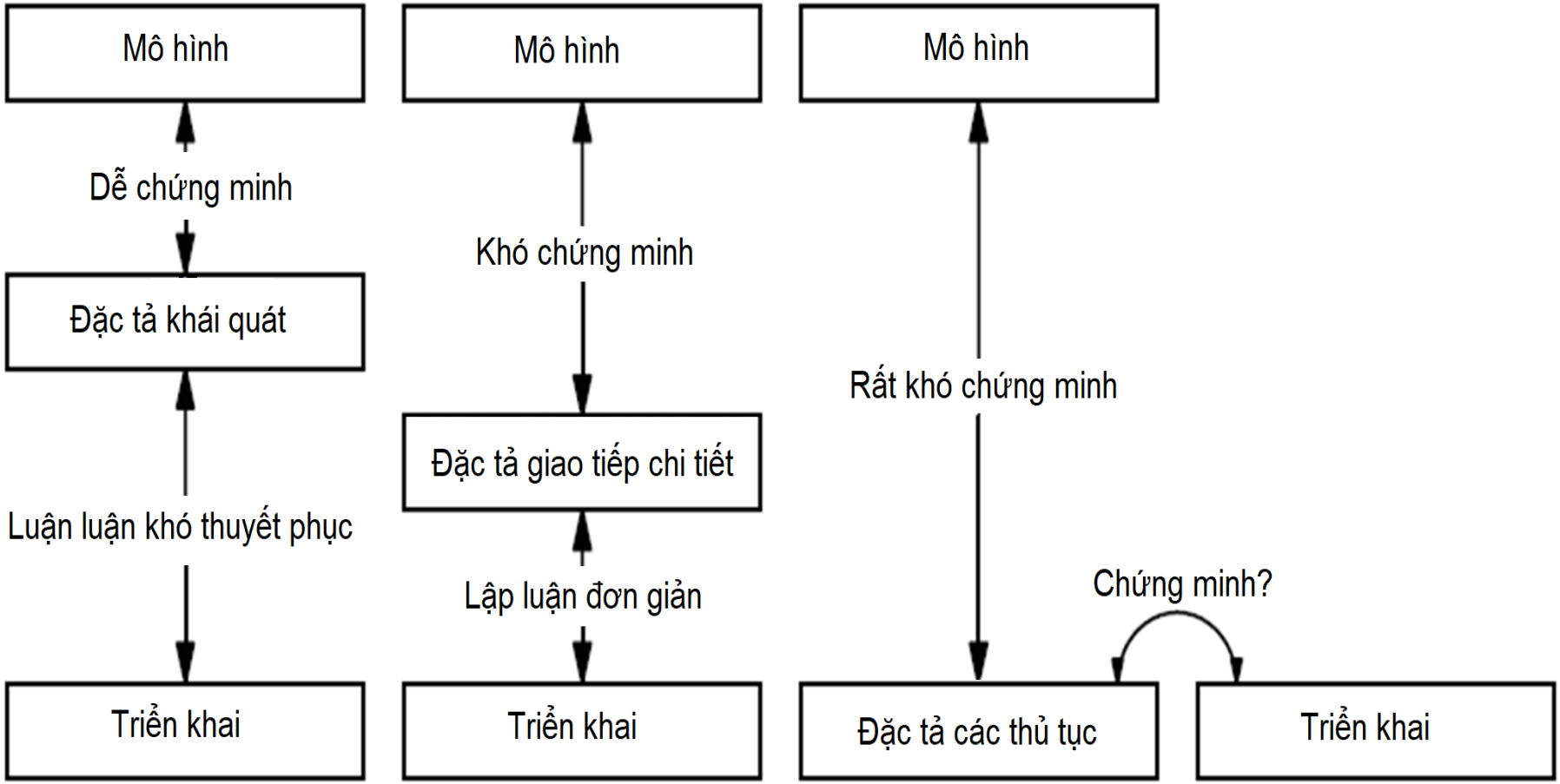
- Ở mức độ tổng quát, các đặc tả an toàn cần khái quát và gần với mô hình an toàn lựa chọn.
  - Như vậy, chúng ta phải đối mặt với khó khăn là diễn giải và chứng minh chi tiết một cách thuyết phục các đoạn mã sinh ra phù hợp với các đặc tả.
- Ở mức độ chi tiết hơn, các đặc tả gần với các thao tác thấy được ở giao tiếp của hệ thống.
  - Đặc tả như vậy sẽ rất phức tạp và khó hiểu và việc chứng minh tương ứng giữa mô hình và triển khai sẽ không thực tế.
  - Mặt khác, các đặc tả biểu diễn các thủ tục/hàm bên trong của hệ thống hơn là các giao tiếp có thể thấy được. Khi đó việc chứng minh mô hình có thể còn khó khăn hơn.

Phân rã dữ liệu và thuật toán - Giới thiệu

- Các đặc tả cung cấp thông tin ở hai mức:
  - Mức trừu tượng gần giống với việc lập mô hình
  - Mức chi tiết mô tả các thao tác hay hoạt động của hệ thống như mô tả các giao tiếp.

Phân rã dữ liệu và thuật toán - Giới thiệu

- Các mức độ chi tiết của việc đặc tả giữa mô hình và việc triển khai



Phân rã dữ liệu và thuật toán - Giới thiệu

- Chứng minh việc triển khai phù hợp mô hình đề ra có thể vô cùng khó khăn, tuy nhiên việc chứng minh mã chương trình phù hợp với đặc tả thì có thể được chấp nhận như việc chứng minh một phần.
- Hai kỹ thuật cơ bản sử dụng cho việc xây dựng các đặc tả chính tắc:
  - Phân rã cấu trúc dữ liệu
  - Phân rã thuật toán.

Phân rã cấu trúc dữ liệu

- Kỹ thuật phân rã dữ liệu (data structure refinement) sử dụng nhiều mức trừu tượng với mức độ chi tiết khác nhau.
- Mỗi lớp đặc tả là một máy trạng thái mô tả hoàn chỉnh hệ thống. Vai trò các lớp như sau:
  - Lớp trên cùng trừu tượng nhất và kết hợp nhiều kiểu dữ liệu, biến và các hàm vào trong một vài hàm đơn giản;
  - Các lớp kế tiếp bổ sung các chi tiết bằng các phân rã các hàm khái quát thành các đối tượng và hàm cụ thể. Các lớp sau là các mô tả cụ thể hơn của hệ thống và thoả mãn các thuộc tính an toàn giống như lớp trên.

Phân rã cấu trúc dữ liệu

- Sau khi hoàn thành lớp sau thì lớp đặc tả trước đó kết thúc vai trò. Lớp dưới cùng rất gần với các biến và hàm trong các đoạn mã của hệ thống.
  - Như vậy, lớp này đảm bảo các mô tả chi tiết và chính xác giao tiếp của hệ thống và giúp cho người thiết kế có thể triển khai được.

❖ Kỹ thuật phân rã cấu trúc dữ liệu không cho biết cách thức thiết kế hệ thống bên trong.

- Việc kiểm chứng cần sử dụng các kỹ thuật công nghệ phần mềm truyền thống như kiểm tra mã nguồn và kiểm thử.

Phân rã thuật toán

- Kỹ thuật phân rã thuật toán (Algorithm refinement) cho phép mô tả một phần cấu trúc nội tại của hệ thống.
- Kỹ thuật này coi hệ thống như một chuỗi các máy trạng thái có phân lớp. Cụ thể:
  - Mỗi máy trạng thái sử dụng các chức năng do lớp trên cung cấp;
  - Việc triển khai các chức năng (function) bao gồm một chương trình khái quát sử dụng các chức năng có trong máy trạng thái ở bên dưới;
  - Lớp thấp nhất cung cấp các chức năng nguyên thủy nhất cả hệ thống mà không thể phân rã thêm được nữa.

Phân rã thuật toán theo các lớp

Lớp	Các đặc tả chính tắc	Các chương trình khái quát
	Giao tiếp với hệ thống ↓	
N	Máy trạng thái mức cao Đặc tả giao tiếp func A func B	**proc** A_N **call** A_{N-1} **call** C_{N-1} **return** **proc** B_N **call** B_{N-1} **call** A_{N-1} **return**
$N-1$	Máy trạng thái trung gian func A func B func C	**proc** A_{N-1} **call** A_{N-2} **call** C_{N-2} **return** **proc** B_{N-1} **call** B_{N-2} **call** A_{N-2} **call** A_{N-2} **return** **proc** C_{N-1} **call** C_{N-2} **return**
$N-2$	Máy trạng thái trung gian	**proc** A_{N-1} **proc** B_{N-1} **proc** C_{N-1}
. . .	. . .	. . .
1	Máy trạng thái trung gian	**proc** A_1 **proc** B_1
0	Máy trạng thái gốc	**proc** A_0 **proc** B_0

Phân rã thuật toán theo các lớp

- Mỗi một bước ứng với một lớp người thiết kế mô tả máy trạng thái giống như trong kỹ thuật phân rã dữ liệu, thêm vào đó, bổ sung chương trình khái quát cho từng chức năng của máy trạng thái.
  - Phần bổ sung này mô tả về thuật toán của các chức năng (hàm) theo dạng lời gọi hàm.

Phân rã thuật toán

- Việc chứng minh đặc tả sử dụng kỹ thuật phân rã thuật toán trước hết cần chứng minh các đặc tả mức cao nhất tương ứng với mô hình xây dựng.
- Tiếp theo, cũng tương tự chứng minh chương trình khái quát của lớp cao nhất phù hợp với đặc tả của nó khi biết các đặc tả các chức năng của lớp kế tiếp.
  - Nhược điểm chính của kỹ thuật này là việc khó thực hiện các chứng minh thuật toán khái quát.

Phân rã thuật toán

- Một số bài toán chứng minh thuộc lớp bài toán khó (intractable).
- Vấn đề khác đó là các đặc tả ở mức cao khá phức tạp do nó biểu diễn giao tiếp thực sự của hệ thống.

❖ Như vậy việc chứng minh mô hình mà hệ thống áp dụng với việc triển khai thực tế sẽ có thể khó khăn.

❖ Tuy vậy, việc này không hạn chế người dùng kết hợp các hai kỹ thuật phân rã dữ liệu và thuật toán để đạt được yêu cầu về đảm bảo tính phù hợp giữa mô hình lựa chọn và việc triển khai thực sự.

Phân rã thuật toán cho thao tác hệ thống file

- Ở mức cao nhất (mức 2) người dùng sẽ chỉ quan tâm tới file và thư mục (directories). Các chức năng căn bản là tạo, xóa file và thư mục và các chức năng giám sát truy cập.
- Mức tiếp theo (mức 1) mô tả các thao tác với file và mô tả file. Mức này đề cập tới cách thức xử lý cấu trúc lưu trữ file trên các thiết bị lưu trữ như làm việc với các thẻ file, các đơn vị cấp phát file. Các chức năng ở mức này vẫn không bị lệ thuộc vào các thiết bị lưu trữ vật lý cụ thể nào.
- Ở mức thấp nhất (mức 0), cách thức làm việc với thiết bị lưu trữ vật lý cụ thể được mô tả. Khi đó, các chức năng sẽ thao tác lên các dữ liệu là các khối cụ thể trên thiết bị lưu trữ.

Phân rã thuật toán cho thao tác hệ thống file

Máy trạng thái khái quát	Các cấu trúc dữ liệu	Các hàm
Machine 2	Files Directories	Create/delete files/directories Read/write files Access control functions
Machine 1	Files File descriptors	Create/delete files Read/write files
Machine 0	Disk blocks	Read/write disk blocks

5.4 Các kỹ thuật kiểm chứng mã chương trình

- Đặt vấn đề
- Phân tích tĩnh
- Phân tích động

Kiểm chứng mã - Đặt vấn đề

- Kiểm chứng, đánh giá mã chương trình nhằm phát hiện và khắc phục các lỗi trong quá trình phát triển phần mềm giảm thiểu các thiệt hại trong quá trình triển khai và vận hành;
  - Các lỗi phổ biến trong quá trình cài đặt mã như tràn bộ đệm, không kiểm tra đầu vào,...
  - Việc xác định các lỗi trong các đoạn mã phần mềm cho phép xác định mức độ an toàn của phần mềm với các lỗi cũng như khả năng đáp ứng của phần mềm với các yêu cầu về an toàn với các đoạn mã trong quá trình xây dựng.
- Quá trình đánh giá mã nguồn thủ công hay tự động đều nhằm mục đích xác định các lỗi liên quan đến an toàn trước khi phần mềm được xuất xưởng.

Kiểm chứng mã - Đặt vấn đề

- Việc đánh giá mã nguồn là cần thiết nhưng chưa đủ để đạt được phần mềm an toàn vì các yếu tố sau:
  - Các lỗi an ninh là các vấn đề khó tránh, song các lỗi thiết kế còn là vấn đề nghiêm trọng hơn.
    - Các lỗi thiết kế hầu như không thể phát hiện thông qua việc đánh giá mã nguồn. Cách tiếp cận đầy đủ để đạt được phần mềm an toàn là kết hợp hài hòa giữa đánh giá mã nguồn và áp dụng quy trình phân tích thiết kế phần mềm an toàn.
  - Việc đánh giá mã nguồn hiển nhiên cần những tri thức về lập trình.
    - Việc hiểu biết các cơ chế an toàn hay an toàn mạng không có ích nhiều với việc đánh giá mã nguồn.
    - Nói cách khác, việc đánh giá mã nguồn tốt nhất là bắt đầu từ người viết ra chúng chứ không phải là các chuyên gia về an toàn.

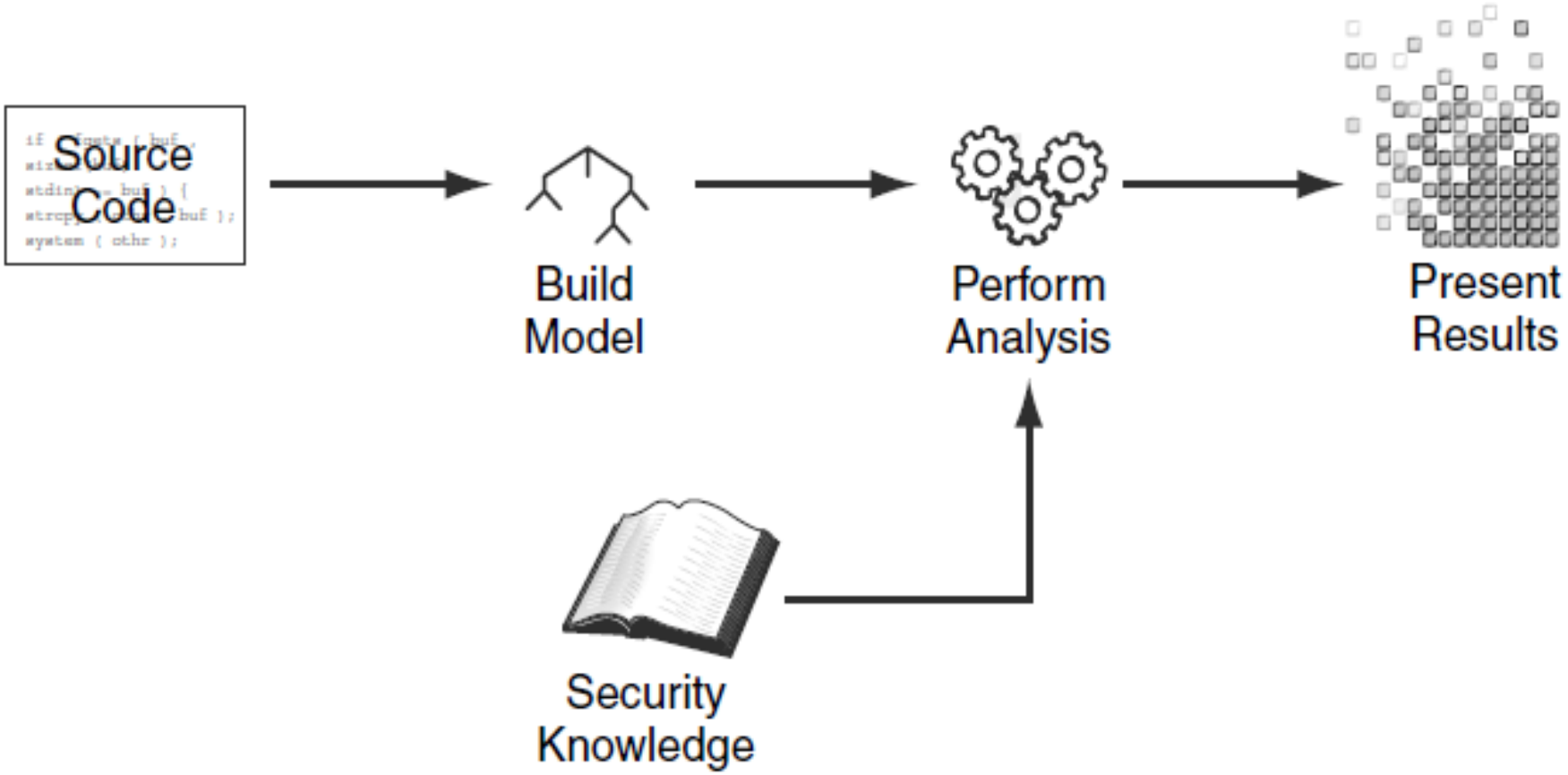
Kiểm chứng mã - Phân tích tĩnh

- Giới thiệu
- Các kỹ thuật phân tích tĩnh
  - Phân tích từ vựng
  - Phân tích câu
  - Phân tích cú pháp khái quát
  - Phân tích ngữ nghĩa
  - Phân tích luồng điều khiển
  - Phân tích luồng dữ liệu
  - Phân tích lan truyền lỗi
- Một số công cụ phân tích tĩnh

Phân tích tĩnh - Giới thiệu

- Phân tích tĩnh đề cập đến các kỹ thuật đánh giá mã nguồn nhằm phát hiện các điểm yếu, lỗ hổng bảo mật tiềm tàng mà không thực thi chúng:
  - Một cách lý tưởng các công cụ tự động có thể tìm kiếm các lỗi bảo mật, tuy nhiên việc này vượt quá khả năng của rất nhiều công cụ hiện thời.
  - Các kỹ thuật kiểm thử phần mềm (testing) thông thường nhằm kiểm tra các hành vi (chức năng) với người dùng thông thường trong điều kiện thông thường nên rất khó để phát hiện ra các lỗi liên quan đến vấn đề an ninh và an toàn.

Mô hình phân tích tĩnh đoạn mã



Mô hình phân tích tĩnh đoạn mã

- Việc đầu tiên công cụ phân tích tĩnh cần làm là chuyển mã chương trình thành mô hình chương trình (program model);
  - Mô hình chương trình thực chất là cấu trúc dữ liệu biểu diễn đoạn mã cần phân tích.
- Việc phân tích được tiến hành kết hợp với các tri thức về vấn đề an toàn biết trước hoặc dựa trên kinh nghiệm.
- Bước cuối cùng là biểu diễn kết quả phân tích theo yêu cầu an toàn đề ra.
  - Dạng cơ bản là cách cảnh báo, các công cụ cao cấp có thể cung cấp các phản ví dụ (khi áp dụng các công cụ dựa trên lô-gíc hay kiểm chứng mô hình).

Phân tích tĩnh - Giới thiệu

- Các kỹ thuật xây dựng mô hình phân tích bao gồm:
  - Phân tích từ vựng (lexical analysis)
  - Phân tích câu (parsing)
  - Cú pháp khái quát (abstract syntax)
  - Phân tích ngữ nghĩa (semantic analysis)
  - Phân tích luồng điều khiển (control flow analysis)
  - Phân tích luồng dữ liệu (data flow analysis)
  - Phân tích lan truyền lỗi (Taint propagation analysis)

Phân tích từ vựng

- Chuyển đoạn mã thành chuỗi các thẻ (token) nhằm loại bỏ những thành phần không quan trọng trong đoạn mã chương trình.
- Các thẻ có thể chứa định danh (biến, tên hàm, ...) và vị trí của chúng trong đoạn mã.
- Đoạn mã dưới đây minh họa cho việc chuyển đổi dòng lệnh ra các thẻ.

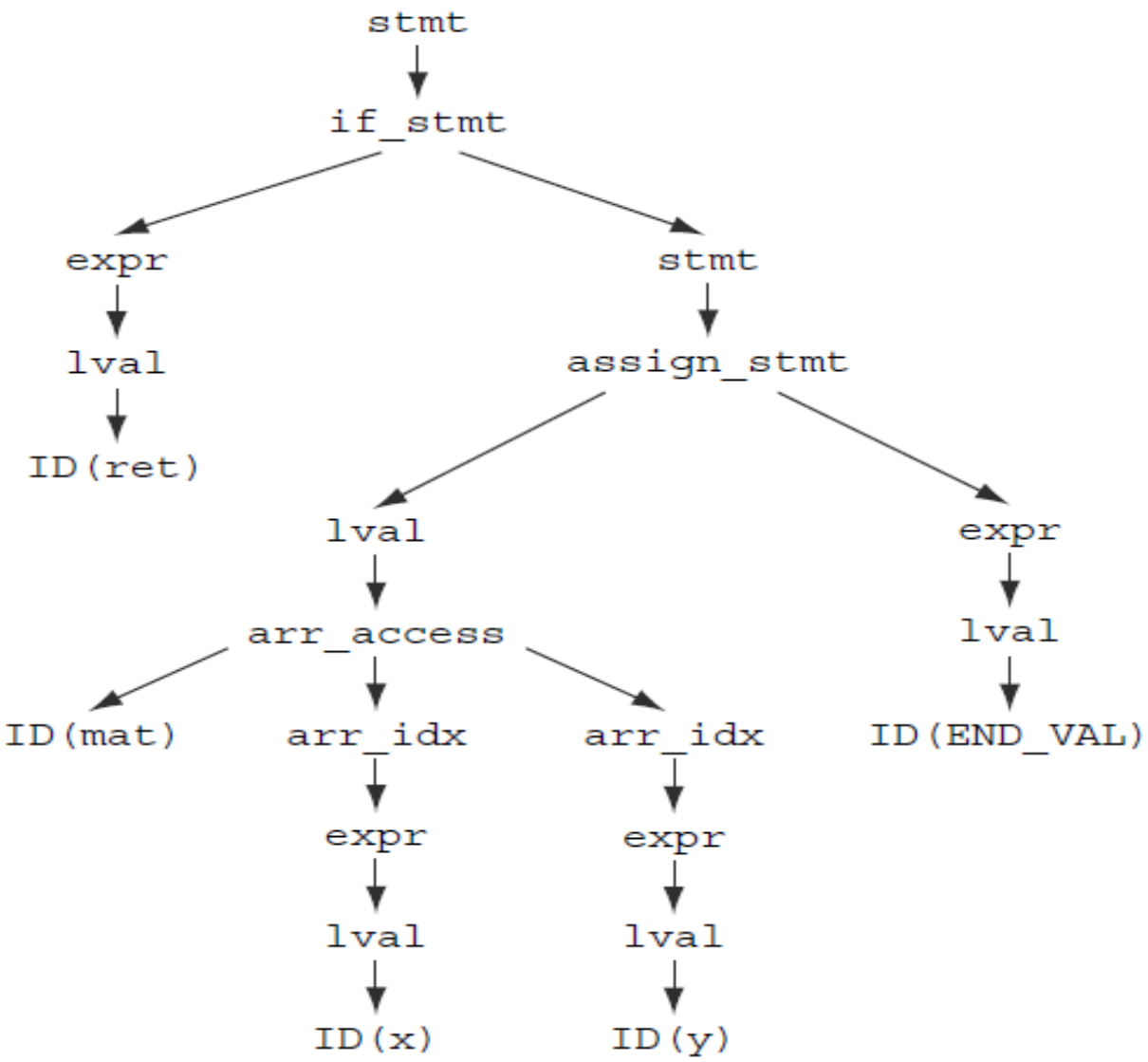
```
if (ret) // probably true
  mat[x][y] = END_VAL;

IF LPAREN ID(ret) RPAREN ID(mat) LBRACKET ID(x) RBRACKET LBRACKET
ID(y) RBRACKET EQUAL ID(END_VAL) SEMI
```

Phân tích câu

- Bộ phân tích câu sử dụng ngữ pháp phi ngữ cảnh (Context-free Grammar-CFG) để đối sánh các chuỗi từ hay thẻ.
  - Bộ ngữ pháp sử dụng các luật sinh để diễn tả các ký hiệu của ngôn ngữ lập trình.
- Ví dụ một số luật sinh cây phân tích

```
stmt := if_stmt | assign_stmt
if_stmt := IF LPAREN expr RPAREN stmt
expr := lval
assign_stmt := lval EQUAL expr SEMI
lval = ID | arr_access
arr_access := ID arr_index+
arr_idx := LBRACKET expr RBRACKET
```

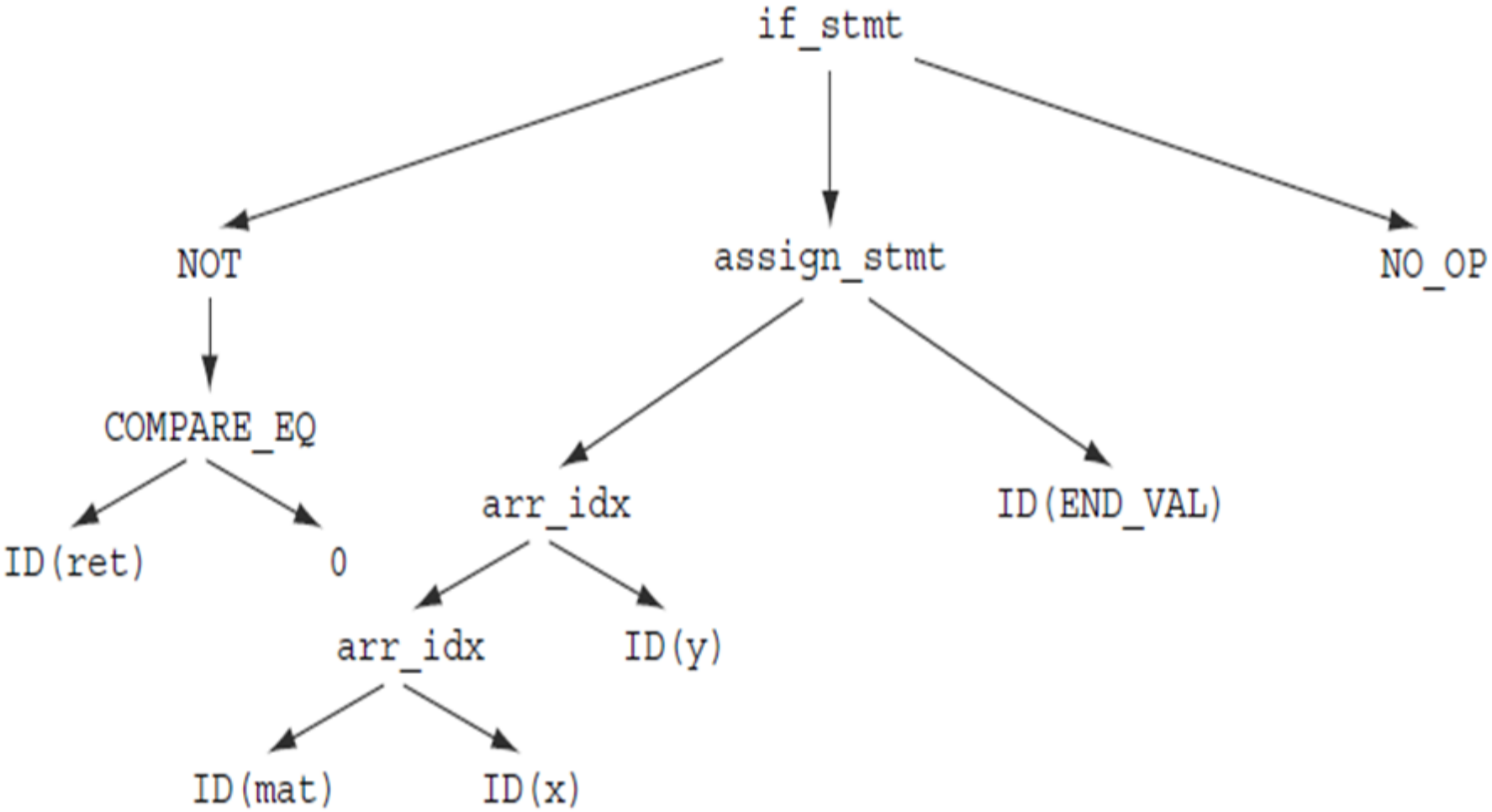
Phân tích câu



Cú pháp khái quát

- Việc phân tích phức tạp có thể không phù hợp trên cây phân tích do mục tiêu của cây phân tích chỉ nhằm vào việc tách từ.
- Vì thế cây cú pháp khái quát cung cấp cấu trúc dữ liệu phù hợp với việc phân tích tiếp theo bằng cách loại bỏ các ký hiệu (thẻ) không phù hợp.
- Các ký hiệu trong cú pháp khái quát có thể ít hơn so với ngôn ngữ lập trình ban đầu.

Cú pháp khái quát



Phân tích ngữ nghĩa

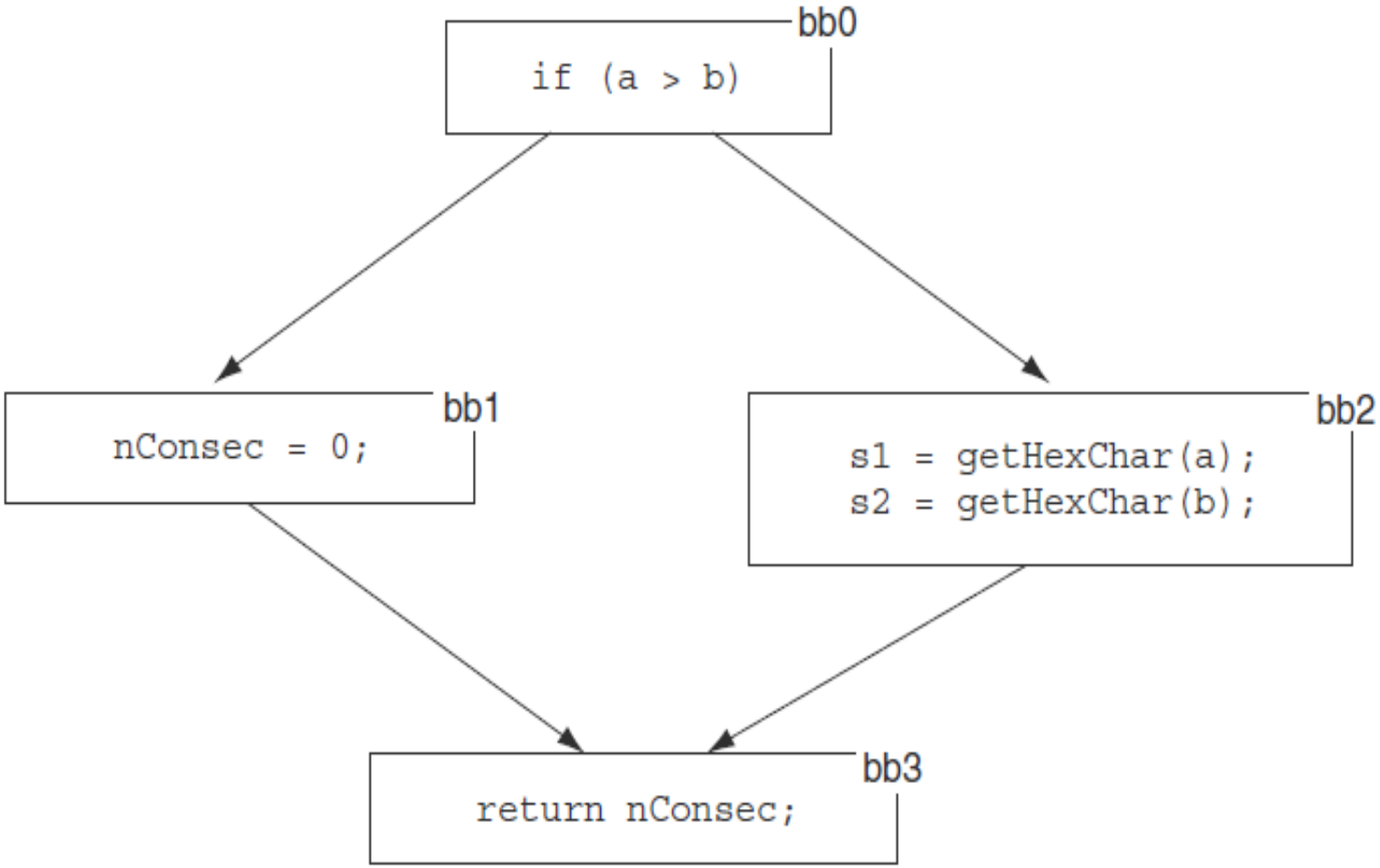
- Việc phân tích ngữ nghĩa có thể được bắt đầu bằng việc phân rã các ký hiệu (tên biến hay hàm) và kiểm tra kiểu dữ liệu.
- Việc phân tích này cho phép lập cấu trúc đoạn mã thông qua việc kiểm tra các lớp của đối tượng được sử dụng, đặc biệt hữu ích với lập trình hướng đối tượng.
- Chức năng phân tích ngữ nghĩa được sử dụng nhiều trong các bộ biên dịch (compiler) của các môi trường phát triển (IDE) của các ngôn ngữ lập trình vì nó cho phép kiểm tra các kiểu dữ liệu, phân rã các tên trong chương trình mô tả các hằng, biến và hàm.

Phân tích luồng điều khiển

- Mục tiêu của kỹ thuật này là để theo dõi các tình huống thực thi khác nhau của đoạn mã và thường được thực hiện bằng cách xây dựng đồ thị luồng điều khiển.
- Việc xây dựng luồng điều khiển giống như việc xây dựng lưu đồ của chương trình từ đoạn mã, như vậy trái ngược với quá trình phát triển phần mềm thông thường (bắt đầu từ xây dựng lưu đồ rồi mới viết đoạn mã).
- Việc tái tạo lại luồng điều khiển cho người phân tích hình dung các khối cơ bản của đoạn mã và cách thức vận hành các khối này.

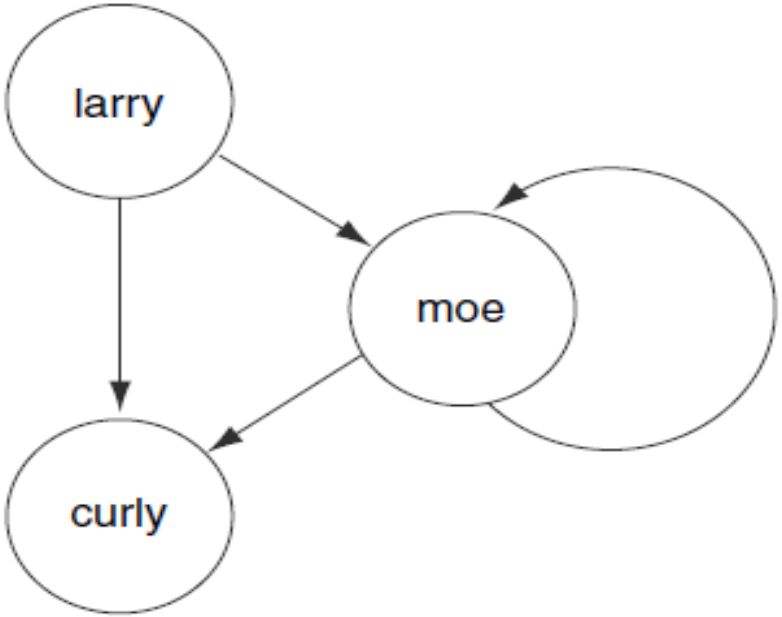
Phân tích luồng điều khiển

❖ Một ví dụ về luồng điều khiển:



Phân tích luồng điều khiển

- Đồ thị gọi hàm biểu diễn luồng điều khiển giữa các hàm hay phương thức và được xây dựng dựa trên đồ thị có hướng.
  - Về cơ bản, đồ thị này thể hiện trạng thái hoạt động của chương trình thông qua việc hàm nào được kích hoạt.
  - Kỹ thuật này thường kết hợp với việc phân tích luồng dữ liệu.
- Đồ thị gọi hàm của ba phương thức larry, moe, curly



Phân tích luồng dữ liệu

- Kỹ thuật phân tích luồng dữ liệu cho phép kiểm chứng cách dữ liệu di chuyển trong đoạn mã.
  - Việc phân tích này thường kết hợp với luồng điều khiển để xác định vị trí bắt đầu và kết thúc của dữ liệu.
  - Việc phân tích luồng dữ liệu có thể phát hiện những tình huống như sử dụng mật khẩu hay khóa cố định (hard-coded) trong đoạn mã.

Phân tích lan truyền lỗi

- Kỹ thuật này được sử dụng để tìm hiểu các giá trị bên trong đoạn mã mà người tấn công có khả năng kiểm soát bằng cách sử dụng luồng dữ liệu.
- Việc này cần thông tin về việc biến chứa lỗi xuất hiện ở đâu trong chương trình và cách thức di chuyển trong chương trình.
- Mục đích của việc phân tích này là để hiểu cách thức các con trỏ có thể tham chiếu đến cùng vị trí nhớ.
- Việc phân tích này rất quan trọng với việc phân tích lan truyền lỗi.

Một số công cụ phân tích tĩnh

Loại công cụ	Địa chỉ
Kiểm tra kiểu lập trình	
PMD	http://pmd.sourceforge.net
Parasoft	http://www.parasoft.com
Kiểm chứng chương trình	
Praxis High Integrity Systems	http://www.praxis-his.com
Escher Technologies	http://www.eschertech.com
Kiểm tra thuộc tính	
Polyspace	http://www.polyspace.com
Grammatech	http://www.gramatech.com
Tìm lỗi	
FindBugs	http://www.findbugs.org
Visual Studio 2005 \analyze	http://msdn.microsoft.com/vstudio/
Đánh giá an ninh	
Fortify Software	http://www.fortify.com
Ounce Labs	http://www.ouncelabs.com

Phân tích động

- Phân tích động phần mềm được thực hiện bằng cách chạy các đoạn mã hoặc cả phần mềm trên bộ xử lý vật lý hay ảo.
- Do các đoạn mã hiện nay thường sử dụng thư viện liên kết động được nạp theo yêu cầu, việc phân tích tĩnh không thể hiện đầy đủ các khía cạnh của đoạn mã.
- Tuy nhiên, việc phân tích động nên được thực hiện sau khi hoàn thành việc phân tích tĩnh do việc thực thi các đoạn mã có thể làm tổn hại đến hệ thống.

Phân tích động

- Trên thực tế, để phân tích và đánh giá tính an toàn của hệ thống, có thể cần thực hiện thêm các phần việc:
  - Quét lỗ hổng
  - Kiểm thử xâm nhập.
- Việc phân tích động có thể được thực hiện thông qua trình gỡ rối debugger. Gỡ rối có thể hoạt động ở mức cao, mức mã nguồn hay mức thấp hơn như mã máy.
  - Các công cụ gỡ rối cung cấp các cơ chế bẫy như chạy từng bước, chạy qua hay lấy thông tin về chương trình tại thời điểm chạy cho phép người dùng kiểm tra hệ quả của từng bộ phận trong chương trình được phân tích.

Phân tích động

- Ưu điểm:
  - Người dùng không phải dự đoán xem trình biên dịch sẽ diễn giải các câu lệnh như thế nào do các câu lệnh của chương trình đã được dịch, Như vậy, việc này loại trừ tính mơ hồ.
  - Mã nguồn chương trình không phải lúc nào cũng có được trong khi mã thực thi hay mã trung gian (như bytecode trong Java) thì lại có sẵn.

Phân tích động

- Nhược điểm:
  - Hiểu được các đoạn mã đã được biên dịch có thể rất khó khăn và một số dạng mã máy rất khó dịch ngược như tập lệnh có độ rộng lệnh thay đổi như Intel x86.
    - Điều này là vì ý nghĩa của đoạn mã thay đổi khi thay đổi vị trí dịch ngược của chuỗi byte biểu diễn mã lệnh.
  - Việc sử dụng mã đã biên dịch khiến cho việc hiểu được ngữ nghĩa (ý định ban đầu của người lập trình) khó khăn hơn nhiều.
    - Đặc biệt khi trình biên dịch thực hiện việc tối ưu hóa đoạn mã thực thi dẫn đến đoạn mã thực thi rất khác so với mã nguồn.