Tạp chí Tin học và Điều khiển học, T. 16, S. 2 (2000), 37–40

MÁY ẢO, CÔNG CỤ HỖ TRỢ HỆ CHẨN ĐOÁN VÀ DIỆT VIRUS TIN HỌC THÔNG MINH

NGUYỄN THANH THỦY, TRƯƠNG MINH NHẬT QUANG

Abstract. Designing an inference engine of an intelligent informatic virus diagnosing and destroying system faces a lot of difficulties, especially in organizing an dentifying enviroment. To make the operation of the inference engine independent of specific computers, we have applied the virtual machine designing techniques in the compiler theory. Thanks to suitable adjustment in function, operation,... our virtual machine supports effectively the inference engine in diagnosing *B*-viruses and opens the prosperity for other kind of viruses.

1. CÁC KHÁI NIỆM

Ngôn ngữ lập trình, trình biên dịch

Lịch sử phát triển của máy tính luôn gắn liền với quá trình phát triển của *ngôn ngữ lập trình* (NNLT). Ngôn ngữ máy với các mã lệnh nhị phân khó nhớ, khó diễn đạt đã dần được thay thế bằng các NNLT khác; từ Hợp ngữ, dạng ngôn ngữ gần gũi với ngôn ngữ máy, đến các NNLT cấp cao tiếp cận phương pháp lập trình cấu trúc, lập trình hướng đối tượng... đã được đề xuất nhằm giúp cho giao tiếp của con người và máy tính dễ dàng, thể hiện được tư duy tự nhiên của con người trước những vấn đề cần giải quyết bằng máy tính. Tuy nhiên, máy tính chỉ có thể thi hành những chỉ thị dưới dạng nhị phân, vì vậy cần có những chương trình dịch những phát biểu, mệnh lệnh của người lập trình (theo những qui định, ngữ nghĩa của NNLT đó) sang dạng mã lệnh mà máy có thể thi hành được. Các chương trình đó được gọi là *trình biên dịch* (TBD).

Càng ngày NNLT càng phát triển với xu thế tách rời những ràng buộc về kiến trúc vật lý của một hệ máy tính cụ thể. Điều này làm cho nhiệm vụ của các TBD càng phức tạp. Mã lệnh mà TBD sinh ra phải thích ứng với nhiều hệ máy tính có bộ chỉ thị, môi trường làm việc (hệ điều hành chẳng hạn) khác nhau. Liệu có thể xây dựng một môi trường trung gian trên cơ sở của một bộ chỉ thị hình thức nào đó? Khái niệm máy ảo (Vitual Machine - VM) bắt nguồn từ những yêu cầu này.

Máy ảo, công cụ hỗ trợ cho các trình biên dịch

Như trên đã nói, lý thuyết TBD được triển khai trên cơ sở của một bộ phân tích cú pháp, ngữ nghĩa và sinh mã lệnh cho chương trình mà chưa hề hay biết gì tới máy mà trình đó sẽ sinh mã cho nó. Để giữ cho việc mô tả TBD được đơn giản, không phụ thuộc đến các tính chất riêng biệt của một bộ xử lý thực đang tồn tại, người ta giả định một máy tính theo chọn lựa riêng và được "gọt giũa" đặc biệt theo yêu cầu của TBD. Đó là một máy giả định, chứ không phải là một bộ xử lý có thật trên thực tế.

Tùy theo yêu cầu của TBD, VM sẽ có cấu trúc, chế độ vận hành phù hợp. Nói chung, một VM bao gồm:

+ *Bộ chỉ thị:* Chứa những chỉ thị hình thức mà trình biên dịch đã định nghĩa dưới dạng bảng tra.

+ *Bộ xử lý lệnh:* Định nghĩa chi tiết về cách thức xử lý của VM dưới dạng một giải thuật. Giải thuật đó lần lượt thể hiện các lệnh của máy.

+ *Bộ nhớ làm việc:* Bộ nhớ làm việc bao gồm:

- Bộ nhớ mã chương trình, được nạp bởi trình biên dịch (đã ở dạng mã máy) và sẽ không thay đổi suốt quá trình thể hiện mã.
- Bộ nhớ dữ liệu, tùy theo nguyên tắc hoạt động của VM mà bộ nhớ này được tổ chức với cấu trúc dữ liệu (CTDL) phù hợp như mảng, ngăn xếp, hàng đợi, danh sách...

Quá trình sinh mã được TBD thực hiện dựa vào bộ chỉ thị hình thức của VM, mã thao tác và tham đối của lệnh. Tham đối này là một số hay một địa chỉ. Các địa chỉ có giá trị là kết quả của phép ánh xạ địa chỉ giữa hai hệ qui chiếu (máy ảo và máy thực) trong một hệ qui chiếu (địa chỉ tương đối so với địa chỉ nơi chương trình được nạp), cùng với các phương pháp tính toán lệnh nhảy, các lời gọi, v.v..

2. MÁY ẢO, CÔNG CỤ HỖ TRỢ HỆ CHẨN ĐOÁN VÀ DIỆT VIRUS TIN HỌC THÔNG MINH

2.1. Vấn đề nảy sinh, cách giải quyết

Quá trình khởi động Hệ điều hành (HĐH) của máy PC được tiến hành sau quá trình POST (Power On Seft Test) bằng việc đọc mẫu tin khởi động (MTKĐ) vào vùng nhớ tại địa chỉ 0:7C00h, sau đó trao quyền cho đoạn mã nằm ở địa chỉ này. Nếu MTKĐ có chứa B-virus, phần khởi tạo (install) của chúng sẽ được kích hoạt và khống chế hệ thống. Trở lại bài toán chẩn đoán B-virus, do MTKĐ chỉ được nạp vào một địa chỉ xác định nên tất cả các địa chỉ tham chiếu có mặt trên MTKĐ đều được xác định *tuyệt đối* từ trước. Vì vậy AntiVirus không thể tự cấp phát một vùng nhớ có đại chỉ tương đối bất kỳ để nạp MTKĐ mà phải sử dụng chính vùng nhớ này để nạp MTKĐ. Có nghĩa là không gian trạng thái cho mô-tơ suy diễn là không gian tĩnh. Đáng tiếc là sau khi hoàn tất quá trình khởi động, HĐH lại sử dụng vùng nhớ này cho mục đích riêng của nó. Qua nghiên cứu các version khác nhau của MSDOS, PCDOS, WINDOWS 3.x, WINDOWS 95, việc sử dụng vùng nhớ này không được HĐH công bố chính thức. Thực tế, chúng dùng chứa các trình điều khiển thiết bị, trình xử lý ngắt, đôi khi được cấp phát cho các ứng dụng, trình thường trú... Vì vậy nếu AntiVirus sử dụng vùng nhớ này, chắc chắn sẽ không tránh khỏi thảm họa sụp đổ toàn hệ thống. Cần tổ chức không gian trạng thái như thế nào để tương thích với tất cả các version của HĐH, mà không phụ thuộc vào môi trường cụ thể của máy lúc AntiVirus* được nạp vào thi hành? Việc nhận dạng hành vi của virus, xét về bản chất đó là sự kết hợp giữa các phương pháp giải quyết vấn đề và xử lý tri thức của Trí tuệ nhân tạo, lý thuyết nhận dạng, cơ chế suy diễn chẩn đoán của Hệ chuyên gia và các kỹ thuật phân tích ngữ pháp, ngữ nghĩa trên ngôn ngữ máy họ 8088, 80×86 của một Trình biên dịch. Liệu chúng ta có thể áp dụng những kỹ thuật đặc trưng của lý thuyết nào để giải quyết vướng mắc? Câu trả lời cho trường hợp này chính là sử dụng kỹ thuật Máy ảo.

2.2. Kiến trúc máy ảo

VM sẽ được thiết kế theo cấu trúc truyền thống [1]. Tuy nhiên để phục vụ tốt cho quá trình chẩn đoán, cũng như tùy thuộc vào đặc điểm của bài toán suy diễn, chúng ta sẽ có một vài hiệu chỉnh cần thiết.

2.2.1. Bộ chỉ thị

VM sẽ phục vụ cho việc nhận dạng virus, là chương trình thực hiện trên máy PC. Vì vậy, bộ chỉ thị của VM phải tương thích với bộ chỉ thị của máy PC dùng bộ vi xử lý 8088, 80×86. Nếu khéo tổ chức, chúng ta có thể tận dụng bộ chỉ thị PC đã được liệt kê trong bản đối chiếu dành cho quá trình nhận dạng lệnh trên cây chỉ thị nhị phân. Tuy nhiên, cần loại bỏ các chỉ thị không phù hợp, ví dụ các lệnh kích hoạt các dịch vụ ở mức hệ điều hành, v.v..

2.2.2. Bộ xử lý lệnh

Chúng ta sử dụng giải thuật xử lý (XL) lệnh trên cây nhị phân để cài đặt bộ XL lệnh cho VM.

Có thể hình dung cơ chế XL lệnh của VM được tổ chức như sau:

(Bộ XL 80×86 $\supseteq$ (Giải thuật XL cây nhị phân) $\supset$ (Bộ XL của VM).

Với cách xây dựng như vậy, ngoài việc tận dụng được các thủ tục đã cài đặt, chúng ta còn thiết lập được mối quan hệ lôgic giữa máy thực (chứa mô-tơ suy diễn) và máy ảo (chứa giải thuật XL lệnh).

2.2.3. Bộ nhớ làm việc

Bộ nhớ cho trình của VM chính là không gian trạng thái của mô-tơ suy diễn trên máy thực. Địa chỉ vật lý của VM sẽ được bố trí tùy theo kích thước vùng nhớ còn trống. Phương pháp định vị địa chỉ tương đối trên hệ qui chiếu VM sẽ do mô tơ suy diễn quản lý, sao cho các công việc định vị địa chỉ vật lý trên máy thực (do B-virus thực hiện) sẽ được ánh xạ thành một địa chỉ tương ứng trên máy ảo. Nhờ vậy, mọi hành vi của B-virus sẽ tái hiện một cách cụ thể trên VM mà không xâm phạm đến môi trường của máy thực. Như vậy việc tổ chức bộ nhớ cho VM rất quan trọng, nó phải thể hiện trung thực tiến trình xử lý lệnh tương thích với bộ XL 80×86 của VM với những đặc trưng cơ bản, các thanh ghi, cờ trạng thái,... Các đại lượng này thực chất là các biến bộ nhớ của AntiVirus*, được khai báo để chứa các giá trị tạm thời trong quá trình vận hành của VM.

Bộ nhớ dữ liệu của VM sẽ được tổ chức với CTDL nào? Khi VM hoạt động các chương trình con đều có thể chứa biến cục bộ và hoạt động một cách đệ qui, nên không thể cấp phát chỗ chứa cho các biến cục bộ đó trước khi chương trình con được gọi đến. Vì vậy, các đoạn dữ liệu cho các trình con được xếp chồng vào một vùng nào đó của bộ nhớ VM trên nguyên tắc LIFO (Last In - First Out – Vào sau ra trước). Nói cách khác, trong bộ nhớ dữ liệu của VM phải có các modul quản lý ngăn xếp làm việc, bao gồm các thanh ghi (biến nhớ) chứa địa chỉ phần tử nằm ở đỉnh ngăn xếp, địa chỉ lệnh đang được thực hiện, địa chỉ trỏ đến lệnh kế sẽ thực thi, các cơ chế định vị địa chỉ trở về...

Ngoài ra, do nhiệm vụ đặc biệt của nó, bộ nhớ của VM cũng phải xác định những địa chỉ đặc tả các vùng hệ thống tương ứng trên máy thực. Vì thế, các đại lượng như bảng véc tơ ngắt, vùng thông tin BIOS chứa địa chỉ cổng nhập xuất, timer... cho VM cũng được tổ chức cho phù hợp.

3. KẾT HỢP GIỮA MÁY ẢO VÀ MÔ-TƠ SUY DIỄN

Như trên đã phân tích, quá trình sinh mã của VM được thực hiện dựa vào bảng tra các chỉ thị máy đã được xây dựng từ trước. Tuy nhiên, khác với VM kinh điển của các TBD, quá trình sinh mã sẽ được tiến hành đồng thời trên cả hai máy ảo và thực. Nhờ các quá trình song song này, ngăn xếp Trace(v) của môtơ (trên máy thực) sẽ quản lý được các hành vi của B-virus trên VM. Kết quả là tiến trình thực hiện của VM sẽ được tái hiện cụ thể trên ngăn xếp Trace(v). Do đó, mô-tơ không phải tốn thêm một giải thuật phân tích lại toàn bộ trình sinh mã. Nếu sau một lần *chạy máy* (ảo), ngăn xếp Trace(v) vẫn giữ giá trị Null như lúc đầu thì chứng tỏ tiến trình nhận dạng thất bại và cho phép kết luận về tính vô nhiễm của chương trình bị nghi vấn có virus [2].

Có thể đánh giá việc thiết kế VM trong quá trình chẩn đoán B-virus trên MTKĐ là một giải pháp heuristics. Thật vậy, nhờ VM này mà mô-tơ suy diễn (trên máy thực) sẽ vận hành một cách trơn tru và nhẹ nhàng. Lúc đó mô-tơ không phải huy động tất cả các luật trên mỗi mã lệnh mà chỉ cần giám sát định kỳ nội dung trên ngăn xếp Trace(v). Ví dụ, sau khi VM thực hiện được n chỉ thị, Trace(v) trả về giá trị đặc tả kích thước vùng nhớ trên VM bị giảm, điều này giúp mô-tơ giả định sự có mặt của B-virus trên VM và chỉ thị *Halt* (dừng thi hành) sẽ được phát đến VM. Saudó mô-tơ sẽ áp dụng các cơ chế suy diễn lùi để kiểm chứng lại các chỉ thị đã được sử dụng cho hành vi tương ứng (*giảm kích thước vùng nhớ*) trên Trace(v). Giải pháp này đồng nghĩa với việc hạn chế hàng loạt các luật dư thừa mà vẫn đảm bảo độ tin cậy cần thiết.

Việc sử dụng VM để thể hiện chiến lược *phân nhiệm* của hệ chẩn đoán virus thông minh. Theo

đó, modul giám sát không cần quan tâm các bộ phận trực thuộc đã thực hiện các công việc (chỉ thị) cụ thể nào, mà chỉ đánh giá công việc thông qua kết quả đạt được, sau một *thời khoản* qui định. Tuy nhiên, để hoàn thành vai trò của mình, mô-tơ phải có thêm các năng lực cần thiết của một supervisor: biết phân công, định thời khoản, biết ra lệnh dừng tiến trình *chạy máy* đúng lúc...

Tất cả những tư tưởng thiết kế, nguyên tắc thực hiện của VM... được trình bày trên đây không đơn thuần là lý thuyết, mà đã qua thực tế cài đặt vận hành. Thật vậy, chúng tôi đã thiết kế một VM có kiến trúc như sau:

- 64 KB RAM.
- 256 vector ngắt 4 byte (đặt tại địa chỉ 0000:0000 của VM).
- 512 byte thông tin điều khiển hoạt động của VM, timer cục bộ...
- Các thanh ghi AX, BX, CX, DX, CS, DS, ES, SS, SP, IP, BP, thanh ghi cờ.
- Bộ chỉ thị tương thích 8088/80×86/Pentium.

Máy ảo này chạy khá tốt. Nó đã góp phần nâng tỷ lệ thành công cho bài toán chẩn đoán B-virus đến 96%, so với tỷ lệ 89% của bài toán chẩn đoán F-virus không sử dụng kỹ thuật VM. Hơn nữa, nhờ quá trình "tinh chế" tệp lệnh trên cây chỉ thị nhị phân (loại bỏ các chỉ thị ở mức HĐH), VM này có thể chạy trên bất cứ HĐH nào (UNIX chẳng hạn) nhưng vẫn cho cùng kết quả như khi chạy trên HĐH mà AntiVirus* sử dụng (MSDOS, PCDOS, WINDOWS 3.x, WINDOWS 95).

Với những ưu điểm nổi bật, liệu chúng ta có thể sử dụng VM cho bài toán chẩn đoán F-virus? Do đặc điểm F-virus ký sinh vào các ứng dụng, được HĐH cấp phát vùng nhớ một cách tương đối, nên không gian trạng thái của mô-tơ suy diễn F-virus là một không gian động. Nói cách khác, việc tổ chức không gian trạng thái trong trường hợp này giống như tổ chức không gian cho một quá trình con đơn thuần. Tuy nhiên, phần install của F-virus và B-virus rất giống nhau nên việc mở rộng khái niệm VM cho trường hợp F-virus là hoàn toàn khả thi. Với những hiệu chỉnh thích hợp nhằm khai thác hiệu quả các chức năng của VM, chắc chắn bài toán nhận dạng F-virus sẽ cho kết quả tốt hơn.

Trong tương lai, hy vọng kỹ thuật VM sẽ được tiếp tục hoàn thiện nhằm tự động hóa quá trình phân tích và lựa chọn giải pháp phục hồi dữ liệu cho tệp bị nhiễm F-virus. Lúc đó, Hệ chẩn đoán và diệt virus thông minh sẽ thể hiện được toàn bộ tiến trình diệt virus của các chuyên gia trong thế giới thực [4].

TÀI LIỆU THAM KHẢO

[1] A. Salomaa, *Nhập môn tin học, Lý thuyết tính toán và các ôtômat*, bản dịch của Nguyễn Xuân My, Phạm Trà Ân, NXB Khoa học và Kỹ thuật, 1992.

[2] N. T. Thuy, T. M. N. Quang, Các cơ chế chẩn đoán virus tin học thông minh, *Tạp chí Tin học và Điều khiển học* **14** (2) (1998).

[3] N. T. Thuy, T. M. N. Quang, Cây chỉ thị nhị phân, không gian tìm kiếm cho việc chẩn đoán virus tin học thông minh, *Tạp chí Tin học và Điều khiển học* **15** (3) (1999).

[4] N. T. Thuy, T. M. N. Quang, A global solution to Intelligent Anti-virus system, *Proceedings of Intelligence Conference on Adcanced Communication Technologies*, Korea, 1999.

[5] N. V. Ba, *Compilateur*, Institut Francophone d'Informatique, 1996.

[6] N. Wirth, *Cấu trúc dữ liệu + Giải thuật = Chương trình*, bản dịch của Hồ Thuần, NXB Thống kê, 1982.

Nhận bài ngày 20 - 11 - 1997

Nguyễn Thanh Thủy - Trường Đại học Bách khoa Hà Nội.
Trương Minh Nhật Quang - Viện Tin học tiếng Pháp.