ویروسهای کامپیوتری: قطری کردن و نقاط ثابت•

ویلیام داولینگ
ترجمهٔ غلامرضا بیات

مقدمه

ویروسهای کامپیوتری به دو دلیل در صدر اخبار کامپیوتری روز قرار گرفته‌اند. اول، به‌دلیل شیوع وسیع ویروسی که درسال ۱۹۸۸ بر تعداد زیادی از کامپیوترهای متوسط در یك شبکهٔ سراسری آمریکا اثر گذاشت. دوم، به‌خاطر تأثیر چند ویروس بر کامپیوترهای شخصی، به‌خصوص از نوع کامپیوترهای آی بی ام و مکینتاش، که به‌طور روزافزون درمراکز کامپیوتری دانشگاهها شایع شده‌اند. درنتیجه، بیشتر افرادی که ازکامپیوترهای شخصی استفاده می‌کنند با نزول فاحشی در عملکرد کامپیوترهای خود مواجه شده‌اند و زمانی را جهت ترمیم خرابی ناشی از ویروس تلف کرده و اعتماد خود را نسبت به‌ارتباطات شبکه‌ای وسیع ازدست داده‌اند. هدف این‌مقاله ارائهٔ یك تعریف ریاضی از ویروس کامپیوتری است (یعنی برنامه‌ای که «تکثیرکنندهٔ خود» است) ونشان دادن اینکه وجود ویروس (از دیدگاه ریاضی) نتیجهٔ اجتناب‌ناپذیر خواص بنیادی هر قامرو محاسباتی است. اگر ازتعریف عملیتری استفاده‌کنیم و بگوییم که ویروس برنامه‌ای است‌که سیستم عاملی را که تحت آن‌کار می‌کند تغییر می‌دهد، دراین‌صورت اگرسیستم عامل مستعد پذیرش ویروس باشد، نمی‌توان بی‌خطر بودن برنامه‌های تشخیص ویروس را تضمین‌کرد. توجه شود که دوتعریف فوق از «ویروس» مجزا و نامعادل‌اند.

تکنیکی که در اینجا برای ساختن ویروس از آن استفاده می‌شود مستقیمترین روش ممکن نیست، ولی من معتقدم که جالب است و پیشنیازی را طلب نمی‌کند. این تکنیك، نمایشی از کاربرد قضیهٔ بازگشتی کلین[1] است و از شکل مثبت برهان قطری استفاده می‌کند. اساس آن بر ایدهٔ کلین دربارهٔ «لم نقطهٔ ثابت قطری» استوار است [۲]. دربارهٔ ماشینهای مولد خود و رابطهٔ آنها با قضیهٔ بازگشتی درمرجع شمارهٔ [۳] شرح داده شده‌است.

1. Kleene

فرمولبندی

چگونه می‌توان ایدهٔ «ویروس»ـ یعنی برنامه‌ای که خودش را تولید می‌کندـ را فرمولبندی‌کرد؟ دراینجا قلمرو برنامه‌سازی یعنی‌زبان برنامه‌سازی و پردازندهٔ آن (نرم‌افزار و سخت‌افزارهای مربوطه) را محدود و مشخص می‌کنیم. فرض‌کنید زبان موردنظر زبان پاسکال باشد ومترجم و سخت‌افزار هم به‌دلخواه شما انتخاب‌شده باشد (انتخاب مهم نیست). برای پرهیز از جزئیات غیرضروری فرض‌می‌کنیم که برنامه‌های پاسکال، ورودیهایشان را ازیك پروندهٔ حرفی می‌گیرند و نتایج خروجی را نیز در یك پروندهٔ حرفی می‌نویسند. اثرات متقابل بین برنامه‌های پاسکال و پردازندهٔ انتخاب شده‌ـ مانند تقسیم برصفر که باعث تولید یك رشتهٔ خاص در پروندهٔ خروجی می‌گردد وغیرهـ را نادیده می‌گیریم. ولی ممکن است برنامه در یك حلقهٔ بی‌انتها قرارگیرد؛ در این حال پردازنده دخالت نخواهد کرد و گفته می‌شود که برنامه نتیجه‌ای به‌دست نمی‌دهد (حتی اگر قبل از اینکه وارد این حلقهٔ بی‌انتها شود مقداری اطلاعات روی پروندهٔ خروجی نوشته باشد). معنی عبارت «P روی ورودی i توقف می‌کند» این است که برنامهٔ P درحین‌پردازش‌ورودی i وارد حلقهٔ بی‌انتها نخواهد شد. بنابراین، این برنامه‌ها مناسبه‌کنندهٔ توابع جزئی از پرونده‌های حرفی به‌پرونده‌های حرفی هستند. مضافاً، بعضی ازاین برنامه‌ها نمایشگر توابعی روی $\mathbf{N}$ (اعداد طبیعی) هستند، یعنی برنامه‌هایی هستندکه ورودی آنها رشته‌ای از ارقام و خروجی آنها نیز رشته‌ای از ارقام است.

فرض کنید $\mathcal{P}$ مجموعهٔ برنامه‌های پاسکال باشد و توجه کنید که $\mathcal{P}$ قابل شمارش است زیرا هر برنامهٔ پاسکال متشکل از رشتهٔ محدودی ازعلامتهایی است‌که ازالفبای محدودی انتخاب‌شده‌اند. اگر مجموعهٔ $\mathcal{P}$ به‌صورت $\{P_0, P_1, \ldots\}$ اندیسگذاری شده باشد، ϕ_i نمایشگر تابعی است (روی پرونده‌های حرفی) که‌توسط برنامهٔ P_i محاسبه می‌شود. $P_i(y)$ نتیجهٔ اجرای P_i روی ورودی

y است و نمایشی است از $\phi_i(y)$ اگر ϕ_i موجود باشد و گرنه تعریف نمی‌شود. همان‌طور که قبلاً گفته شد، بعضی از ϕها را می‌توان به‌صورت توابعی روی $\mathbf{N}$ ناقی نمود، بنابراین گاهی عباراتی نظیر $\phi_{\phi_i(j)}$ نیز مورد استفاده قرار می‌گیرند. اگر $\phi_i(j)$ مشخص‌کنندهٔ یک عدد طبیعی نباشد، $\phi_{\phi_i(j)}$ به‌یک تابع کاملاً نامعین اطلاق می‌شود که نتیجهٔ محاسبهٔ برنامهٔ پاسکالی خواهد بود که با هر ورودی درحلقهٔ بی‌انتها قرار می‌گیرد.

دراینجا ماهیت دقیق اندیسگذاری چندان موردنظر ما نیست، بجز اینکه لازم است این اندیسگذاری به‌صورتی انجام گیرد که محاسبهٔ توابع زیر توسط یک برنامهٔ پاسکال امکان‌پذیر باشد:

۱. $Rep: i \mapsto P_i$

۲. $Comp: \mathbf{N}\times\mathbf{N} \to \mathbf{N}$. با معلوم بودن اندیسهای دو برنامهٔ P و Q، تابع $Comp$ اندیس ترکیب دوبرنامهٔ P و Q را محاسبه می‌کند یعنی برنامه‌ای‌است که ورودیش را به Q می‌دهد و اگر وهنگامی‌که Q توقف کند P را روی نتیجهٔ خروجی Q اجرا می‌کند.

۳. یک مقلد پاسکال، که دومتغیر P و i را به‌عنوان ورودی قبول می‌کند؛ P نمایشگر یک برنامهٔ پاسکال به‌صورت رشته‌ای از علامتهاست. مقلد پاسکال همان نتیجه (یا حلقه)ای را به‌دست می‌دهد که برنامهٔ P روی ورودی i می‌دهد.

فهرستی الفبایی از تمام برنامه‌های پاسکال دراین شرایط صدق می‌کند (اگرچه ممکن است تصور شود که اجرای توابع Rep و $Comp$ زمان بسیار زیادی را می‌طلبد).

با معلوم بودن سیستم اندیسگذاری و تابع Rep، می‌توان مشخص کرد که برنامه‌ای‌که خودش را تولید می‌کند چیست.

تعریف. ویروس برنامه‌ای است چون $P=P_{\_}$ به‌طوری که برای هر ورودی y داشته باشیم $P(y)=Rep(m)$.

حال به‌نحوهٔ ساخت ویروس می‌پردازیم.

قطری‌کردن و یک قضیهٔ نقطهٔ ثابت

در نظر بیشتر ریاضیدانان، برهانهای مبتنی بر قطری‌کردن برای به‌دست‌آوردن نتایج منفی به‌کارمی‌روند. دوقضیه که به‌قضیهٔ کانتور معروف‌اند و به‌روش قطری‌کردن اثبات می‌شوند، می‌گویند «تمام اعداد حقیقی دربازهٔ $[0,1]$ نمی‌توانند به‌صورت مجموع نامتناهی $\sum_{i=1}^{\infty} b_i 2^{-i} (b_i \in \{0,1\})$ ظاهر شوند» و «عدد اصلی هیچ مجموعهٔ Sی برابر با عدد اصلی مجموعهٔ توانی S نیست.» قضیهٔ منفی دیگری که به‌روش قطری‌کردن ثابت می‌شود عبارت است ازاینکه «برنامهٔ پاسکالی وجود ندارد که بتواند مسألهٔ عضویت درمجموعهٔ $\{$برنامهٔ P_i روی‌ورودی i توقف می‌کند$/i \in \mathbf{N}\}$ را معلوم کند.» عصارهٔ برهان مبتنی برقطری کردن به‌صورت زیر است: اگر M_{ij} ماتریسی مربعی (حتی نامتناهی) باشد، هیچ سطر M وجود ندارد که تمام درایه‌هایش متمایز ازقطر M باشد، زیرا درایهٔ i ام سطر i با درایهٔ iام قطر M مطابقت می‌کند. نتایج منفی‌فوق با نشان‌دادن اینکه خلاف آنها وجود یک ماتریس مربعی را نتیجه می‌دهد که یک سطر آن از قطر ماتریس کاملا متمایز است، قابل اثبات‌اند. به‌این‌ترتیب «پارادوکسهایی» به‌وجود می‌آید: اگرسلمانی b وجود داشته باشد که وظیفه‌اش اصلاح تمام افرادی باشد که نمی‌توانند خود را اصلاح کنند، می‌توان ماتریس M را به‌صورت زیر تعریف کرد

$$M_{ij}=\begin{cases}1 & i,j \text{ را اصلاح می‌کند}\\ 0 & \text{درغیراین صورت}\end{cases}$$

و سطر M_b قرینهٔ قطر ماتریس است به‌این معنی که به‌ازای هر j، $M_{bj}=1-M_{jj}$. پس چنین سلمانی وجود ندارد.

روش قطری‌کردن را به‌صورت مثبت نیز می‌توان به‌کاربرد و یک‌قضیهٔ نقطهٔ ثابت به‌دست‌آورد [۲]. فرض‌کنید f یک‌تابع باشد. گویند ماتریس مربعی M دقت f سطر به‌سطر بسته است اگر هر سطر آن به‌وسیلهٔ تابع f به‌سطر دیگری وابسته باشد، یعنی به‌ازای هر i یک k وجود داشته باشد به‌طوری که برای هر j، $f(M_{ij})=M_{kj}$. به‌علاوه M به‌طور قطری کامل است اگرسطری از M با قطر M یکی باشد. جالب است که وجود Mی با ویژگیهای دوگانهٔ فوق تضمین‌کنندهٔ این مطلب است که f دارای نقطهٔ ثابت است. برای اثبات این مطلب، فرض‌کنید سطر M_i مساوی قطر M و سطر k تصویرسطر i تحت f باشد. دراین‌حال، $f(M_{ik})=M_{kk}=M_{ik}$ یعنی M_{ik} تحت f ثابت است، و این «لم نقطهٔ ثابت قطری» است.

حال فرض کنید $f: \mathbf{N} \to \mathbf{N}$ تابعی است که توسط یک برنامهٔ پاسکال محاسبه می‌شود و ضوابط مورد نظر ما را در اندیسگذاری برنامه‌های پاسکال دارد، یعنی

$$\phi_i=\phi_j \Rightarrow \phi_{f(i)}=\phi_{f(j)}.$$

در این حالت f را «توسیعی» خوانند. چون f توسیعی است، تعیین‌کنندهٔ یک عملگر روی توابع ϕ است (که آن هم با f نشان داده می‌شود) به‌طوری‌که $f(\phi_i)=\phi_{f(i)}$. با استفاده از لم نقطهٔ ثابت قطری نشان می‌دهیم یک $n \in \mathbf{N}$ وجود دارد به‌طوری‌که $\phi_n=\phi_{f(n)}$. فرض‌کنید M یک‌ماتریس نامتناهی ازتوابع جزئی باشد: $M_{ij}=\phi_{\phi_i(j)}$. نیز توجه کنید که هرعنصر M_{ij} را می‌توان توسط یک برنامهٔ پاسکال P_i محاسبه کرد، ودرواقع چنین Iای را می‌توان از i و j به‌دست‌آورد. P_I برنامه‌ای است که: (۱) با استفاده‌از Rep، P_i را می‌سازد. (۲) با استفاده ازیک زیربرنامهٔ مقلد پاسکال، عمل P_i روی ورودی j را مشخص می‌کند. (۳) اگر و هنگامی که مرحلهٔ ۲ تمام شود با استفادهٔ مجدد از Rep، $P_{\phi_i(j)}$ را می‌سازد و (۴) با استفادهٔ مجدد از مقلد پاسکال عمل برنامهٔ ساخته‌شده درمرحلهٔ ۳ را روی ورودیش (P_I) محاسبه می‌کند.

لازم‌است‌ثابت کنیم M به‌طورقطری‌کامل وتحت f بسته است. فرض‌کنید P یک برنامهٔ پاسکال باشدکه روی هر ورودی $j \in \mathbf{N}$ همان‌چیزی را محاسبه‌کندکه برنامهٔ $P_j(j)$ می‌کند. (همان‌طورکه قبلاً گفته شد این مستلزم فراخوانی Rep و فراخوانی مقلد پاسکال است). فرض‌کنید k اندیس برنامهٔ P باشد، پس به‌ازای هر $j \in \mathbf{N}$ داریم $\phi_k(j)=\phi_j(j)$ بنابراین $\phi_{\phi_k(j)}=\phi_{\phi_j(j)}$ و سطر k از M برابر قطر است. حال نشان می‌دهیم که M تحت

f بسته است. فرض کنید P یک برنامهٔ پاسکال (وابسته به i) باشد که ورودی j را قبول کرده و مراحل زیر را طی می‌کند:

۱. $Rep(i)$ را محاسبه می‌کند.

۲. از مقلد پاسکال استفاده‌کرده نتیجهٔ مرحلهٔ ۱ را در مورد ورودی j به‌کار می‌گیرد.

۳. اگر و هنگامی که مرحلهٔ ۲ تمام شود، تابع f را روی نتیجهٔ این مرحله محاسبه می‌کند.

فرض کنید $k=k(i)$ اندیس برنامهٔ P باشد. بنابراین بجز در حالتی که $\phi_i(j)$ در حلقهٔ بی‌انتها قرار گیرد، داریم $\phi_k(j)=f(\phi_i(j))$ که در این حالت هردو عبارت نامعین‌اند. بنابراین برای هر j داریم:

$$f(M_{ij})=\phi_{f(\phi_i(j))}=\phi_{\phi_k(j)}=M_{kj}$$

یعنی M تحت f سطربه‌سطر بسته‌است. ازاین‌واقعیت‌که f توسط برنامهٔ پاسکال قابل محاسبه است در محاسبهٔ k استفاده شده و توسیعی‌بودن f برای‌معنی‌داربودن‌عبارت $f(M_{ij})$ لازم‌است. تحت این‌شرایط برای f و فرضهای قبلی دربارهٔ اندیسگذاری، می‌توان از لم نقطهٔ ثابت قطری استفاده‌کرد و شکل ضعیف شده‌ای ازقضیهٔ بازگشتی را به‌صورت زیر بنا نهاد. n ی متعلق به $\mathbf{N}$ وجود دارد به‌طوری‌که $\phi_n=\phi_{f(n)}$. توجه داشته‌باشیدکه n به‌شیوهٔ ساختنی[1] از (یک برنامه برای، یا اندیس) f و فقط با استفاده از توابع $Comp$، Rep و مقلد پاسکال به‌دست می‌آید.

قضیهٔ بازگشتی را می‌توان قویتر کرد تا شرط توسیعی‌بودن f قابل‌حذف باشد. فرض‌کنید f تابعی‌دلخواه روی $\mathbf{N}$ وقابل‌محاسبه با پاسکال‌باشد. گوییم ϕ_i و ϕ_j تحت f وابسته‌اند اگر kیی وجود داشته‌باشد به‌طوری‌که $\phi_i=\phi_k$ و $\phi_j=\phi_{f(k)}$. سپس برهان‌فوق نشان می‌دهدکه هرسطر i از M تحت f به‌سطری چون k از M وابسته است (شیوهٔ ساختن k همانند بالاست). چون قطر M یکی ازسطرهای M است، عنصری قطری چون (M_{ii}) تحت f وابسته به‌خودش است. بنابراین kیی وجود داردکه $\phi_k=M_{ii}=\phi_{f(k)}$.

حال نشان می‌دهیم یک‌ویروس، به‌شکلی‌که دربالا تعریف شد، چگونه ساخته می‌شود. فرض‌کنید r یک اندیس برای Rep باشد، و نیز فرض‌کنید $f:\mathbf{N}\to\mathbf{N}$ تابعی باشد که به‌ازای هر اندیس x، اندیس یک برنامهٔ پاسکال P را، که عملیات زیر را انجام می‌دهد، برمی‌گرداند: (۱) مقدار $Comp(r,x)$ را محاسبه می‌کند. (۲) مقدار محاسبه شده‌را مستقل از ورودی برنامهٔ P برمی‌گرداند (بنابراین f همواره اندیس یک برنامه را محاسبه می‌کندکه‌کار آن برنامه، محاسبهٔ یک تابع ثابت است، اما تابع ثابت بستگی به‌مقدار x دارد.) روشن است که f توسط یک برنامهٔ پاسکال محاسبه می‌شود، با فرض اینکه $Comp$ مربوط به‌آن وجود داشته باشد. (ولی توجه‌کنید که f توسیعی نیست). بنابه‌قضیهٔ بازگشتی n ی وجود دارد به‌طوری‌که $\phi_n=\phi_{f(n)}$ که تابعی از y است و مقدار $Comp(r,n)$ را برمی‌گرداند. بنابراین:

به‌ازای هر y، $\quad Rep(\phi_n(y))=Rep(Comp(r,n))$

به‌ازای هر y، $\quad P_{Comp(r,n)}(y)=Rep(Comp(r,n))$

ویروس، برنامه‌ای است با اندیس $Comp(r,n)$: روی هرورودی y خروجی آن نمایشی است از خودش به صورت رشته‌ای از حروف.

تمامی مراحل توصیف شده در بالا قابل به‌کارگیری‌اند و علی‌الاصول می‌توان برنامه‌های ویروس‌کامپیوتری را به‌همین طریق نوشت. ولی البته روش نوشتن ویروسها عملا این طور نیست. با این حال، روش فوق نشان می‌دهدکه می‌توان وجود برنامه‌هایی را که مولد خود هستند تصورکرد. دلیلش این‌است‌که فقط‌فرضیات خیلی ضعیفی در اثبات وجود ویروس زبان پاسکال شده بنود: هرزبان برنامه‌سازی که بتوان برای آن تعبیرکننده‌ای نوشت، و بتواند برنامه‌های $Comp$ و Rep را اجرا کند مستعدپذیرش ویروس است. مفهوم اصلی درساخت ویروس، یک تابع جانشانی (مانند f) است، که ازنقطهٔ ثابت آن برای تولید ویروس استفاده می‌شود. در برنامه‌های عملی ویروس [۴] از تابع جانشانی و یک نقطهٔ ثابت آن، همان‌طورکه تابع f و یک نقطهٔ ثابت آن دراینجا به‌کار رفت، استفاده می‌کنند ولی نحوهٔ ساخت آنها به‌مراتب ساده‌تر است.

تشخیص ویروس: قطری‌کردن منفی

حال می‌پردازیم به موضوعی ملموستر. برنامه‌هایی که به‌وسیلهٔ کامپیوترهای مدرن اجرا می‌شوند، برخلاف آنهایی‌که تا اینجا توصیف شده‌اند، «دریک محیط» اجرا می‌شوند یعنی وقتی برنامه‌ای اجرا می‌شود، به‌عنوان زیربرنامه‌ای از یک برنامهٔ دیگر، موسوم به‌سیستم عامل، که از برنامهٔ ما استقلال منطقی دارد اجرا می‌شود. وظیفه سیستم‌عامل، مدیریت حافظهٔ اصلی وفرعی، مدیریت‌پردازنده، نگهداری آمارهای مربوطه و نظایر آن است. از لحاظ عملی ویروسها، آن طور که در بالا توصیف شدند، «سیستم عامل را تخریب می‌کنند». یعنی کپی‌هایی ازخودشان را درسیستم عامل پدید می‌آورند.

برای نشان دادن این اثر تخریبی، در بقیهٔ مقاله از تعریف جدیدی از ویروس استفاده خواهیم کرد.

تعریف. ویروس، برنامه‌ای است‌که وقتی اجرا می‌شود، کد سیستم عامل را تغییر می‌دهد.

(وقتی نسخهٔ جدیدی به‌صورت مجاز ودرست از سیستم عامل تهیه می‌شود، به‌جای سیستم عامل قدیم قرار می‌گیرد، نه‌اینکه آن را تغییر دهد.) توجه داشته باشید که لازم نیست سیستم عامل تغییریافته رفتار خاصی داشته باشد، یعنی ازنظر نتایج مورد علاقهٔ ما تعیین شرایط باز تولید سیستم عامل لازم نیست.

خوب بود اگر می‌توانستیم به‌طور خودکار و از طریق یک برنامهٔ تشخیص‌دهندهٔ ویروس، تشخیص دهیم که‌کدام برنامه‌ها ویروسی‌اند وکدام برنامه‌ها نیستند. درآن صورت ازصرف هزینه و دردسری که از تغییر ناخواسته و احتمالا زیان‌آور سیستم عامل ناشی‌می‌شود جلوگیری‌می‌کردیم. حال‌نشان می‌دهیم هیچ برنامه‌ای،

1. constructive

قادر به‌تشخیص صحیح ویروس برای هر ورودی ممکن نیست و هیچ تضمینی برای جلوگیری از پخش ویروسها وجود ندارد.

کار را با انتخاب یك سیستم عامل مشخص OS و با تعریف زیر شروع می‌کنیم.

تعریف. گوییم برنامهٔ P ویروسی را روی ورودی x پخش می‌کند اگر با اجرای برنامهٔ P تحت سیستم عامل OS و روی ورودی x، OS تغییر کند. در غیر این صورت گوییم P روی ورودی x بی‌خطر است. برنامه‌ای را بی‌خطر گویند که برای هر ورودی بی‌خطر باشد. و نیز فرض می‌کنیم که برای OS ویروسهایی وجود دارد، (در غیر این صورت نیازی به آزمایش نیست).

حال‌از برهان‌خلف استفاده می‌کنیم. فرض‌کنید برنامهٔ بی‌خطری به‌نام $IS\text{-}SAFE$ وجود دارد که بی‌خطر بودن یا نبودن اجرای هر برنامهٔ اختیاری P برای هر ورودی اختیاری x را معلوم می‌کند. بنابراین

$$IS\text{-}SAFE(P,x)=\begin{cases}\text{yes} & \text{اگر } P \text{ روی ورودی } x \text{ بی‌خطر باشد}\\ \text{no} & \text{درغیراین‌صورت}\end{cases}$$

با توجه به‌برنامهٔ فوق و این فرض که ویروسی وجود دارد، نوشتن برنامهٔ $D()$ از یك متنیر که دارای رفتار زیر است، ساده است.

$$D(P)=$$

$$\begin{cases}\text{Write“Have a nice day”}, & IS\text{-}SAFE(P,P)=\text{no} \text{ اگر}\\ \text{OS تغییر داده شود} & \text{درغیراین‌صورت}\end{cases}$$

برنامهٔ D از قطر ماتریس $M_{P,x}=IS\text{-}SAFE(P,x)$ کاملا متمایز است.

حال با بررسی رفتار برنامهٔ D روی ورودی D می‌توانیم نشان دهیم که $IS\text{-}SAFE$، نمی‌تواند هم بی‌خطر باشد و هم درست. اگر D روی ورودی D بی‌خطر باشد فقط به‌خاطر این است که قسمت«درغیراین‌صورت» را اجرا نکرده‌است، یعنی چون $IS\text{-}SAFE(D,D)=\text{“no”}$، پس معلوم می‌شود $IS\text{-}SAFE$ درست نیست. از طرف دیگر اگر D، OS را روی ورودی D تغییر دهد دو امکان وجود دارد. از یك طرف نتیجهٔ فراخوانی $IS\text{-}SAFE(D,D)$ ممکن است "yes" باشدکه در این حالت قسمت«درغیراین‌صورت» برنامهٔ D اجرا شده‌است، که‌دراین‌حال $IS\text{-}SAFE$ درست نیست. از طرف دیگر اگر نتیجهٔ فراخوانی $IS\text{-}SAFE(D,D)$ مقدار "no" باشد (دراین‌صورت D فقط "Have a nice day" را چاپ می‌کند) فرض اینکه D روی ورودی D بی‌خطر باشد بدین‌معنی است که فراخوانی $IS\text{-}SAFE$ بایستی مقصر باشد، یعنی $IS\text{-}SAFE$ بی‌خطر نیست. ما به‌این نتیجه رسیدیم که فرض وجود یك برنامهٔ بی‌خطر و درست، که روی OS اجرا می‌شود، و بی‌خطر بودن ورودی خود را آزمون می‌کند، غلط است، و برنامه‌ای مانند $IS\text{-}SAFE$ نمی‌تواند وجود داشته باشد.

فرض وجود ویروس برای OS، که در اثبات عدم وجود $IS\text{-}SAFE$ لازم است، برای هر سیستم عامل صحیح نیست. یك سیستم عامل، و حتی بعضی از برنامه‌هایی که تحت کنترل آن اجرا می‌شوند، ممکن است درقسمتی ازحافظهٔ‌کامپیوتر ذخیره شود که نتوان چیزی روی آن نوشت. دراین صورت هیچ برنامه‌ای وجود ندارد که اجرای آن باعث تغییر سیستم عامل گردد، و بنابراین یك‌برنامهٔ‌بی‌خطر $IS\text{-}SAFE$ وجود داردکه همواره‌تابع ثابت "yes" را محاسبه می‌کند. مثلا بعضی کامپیوترهای شخصی سیستم عامل خود را در ROM (حافظهٔ فقط خواندنی) نگهداری می‌کنند. در سیستمهای محاط شده (مثلا کامپیوترهای کنترل کننده VCR ها و سیستمهای احتراق اتومبیل) هم سیستم عامل و هم برنامه‌هایی‌که تحت‌کنترل آن کار می‌کنند در ROM ذخیره می‌شوند. می‌توان چنین حالاتی را به‌وسیلهٔ یك برنامهٔ سیستم‌عامل نشان دادکه تابعی باشد که قلمرو و برد آن عناصر سه‌تایی‌اند و به‌صورت $\langle P,x,OS\rangle \mapsto \langle P,P(x),OS\rangle$ تعریف می‌شود. سیستم عامل نه تنها $P(x)$ را محاسبه می‌کند، بلکه OS و P را هم (بدون تغییر) برمی‌گرداند. ملاحظاتی نظیر آنچه در بخش پیش‌گفته شد منتهی به‌این می‌شودکه برنامه‌هایی وجود دارندکه چنین تابعی را محاسبه می‌کنند. طبق‌تعریف، این برنامه‌ها تخریب نمی‌شوند. (به‌خاطر اینکه اصل سیستم عامل، و نه نسخهٔ تغییریافتهٔ آن، را برمی‌گردانند).

ولی درکامپیوترهای هدف‌منظوره، هم سیستم عامل و هم‌برنامهٔ P در ناحیهٔ فیزیکی مشترکی از حافظه اصلی کامپیوتر (حافظه مجازی یا حقیقی) قرار دارند و این نوع حافظه‌ها نوشتنی‌هستند. اجرای برنامهٔ P، حتی تحت‌کنترل‌سیستم عامل، امکان دارد محلی از حافظه را که سیستم عامل در آن ناحیه قرار دارد تغییر دهد. به‌علاوه، به‌دلایل عملی، سیستمهای عامل در واقع خودشان را در هر اجرای برنامه نمی‌نویسند، بلکه فقط دومین مؤلفه از سه‌تایی فوق را محاسبه می‌کنند. این ترکیب عوامل، وجود ویروس را امکان‌پذیر می‌سازد و فرضی را که منتهی به‌عدم وجود $IS\text{-}SAFE$ می‌شود تأیید می‌کند.

مراجع

1. Dowling, William F., "There are no safe virus tests", *Am. Math. Monthly*, 96.9, pp. 835-6.
2. Owings, J.C., "Diagonalization and the recursion theorem", Notre Dame Journal of Formal Logic, Vol. 14, Number 1,1973, pp. 95-99.
3. Rogers, H., *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, 1967.
4. I. Witten, Computer (in) security: infiltrating open systems, *Abacus* 4 (Summer 1987) 7-25.

****

• William F. Dowling, "Computer viruses: diagonalization and fixed points," *Notices of the American Mathematical Society* (7) 37 (1990) 858-861.