计算机病毒的计算复杂度问题

左志宏[1] 舒 敏[2] 周明天[1]
(电子科技大学计算机科学与工程学院 成都610054)[1]
(国家计算机网络与信息安全管理中心 北京100032)[2]

摘 要 计算机病毒对计算机系统及软件造成各种各样的损害,除了一些常见的损害,例如删除数据或程序、修改系统信息以外,它们还造成一些非破坏性的影响,例如消耗大量的存贮及时间。这个问题涉及到计算机病毒的计算复杂度。文章初步探讨计算机病毒的计算复杂度问题,从数学上证明两个基本结论:存在计算机病毒,它的传染过程具有任意大计算复杂度;存在计算机病毒,被感染程序的执行过程具有任意大的计算复杂度。除此而外,文章简要讨论计算机病毒检测过程的计算复杂度问题。

关键词 计算机病毒,计算复杂度,损害,传染,模拟

Computational Complexity of Computer Viruses

ZUO Zhi-Hong[1] SHU Min[2] ZHOU Ming-Tian[1]
(School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054)[1]
(Management Center of National Computer Network and Information Semrity, Beijing 100032)[2]

Abstract Computer viruses make various injuries on computer systems or software. Beside some familiar injuries, such as destroying on data or programs, modifying system information, etc., they also take some nondestructive effects such as consuming large amount of storage or time. This problem is pertinent to computational complexity of computer viruses. This paper takes initial step to some issues on computational complexity about computer viruses, and gives mathematical proofs of two basic results, which says that there exist computer viruseswith arbitrary computational complexity in their infecting procedure and computer viruses with arbitrary computational complexity in their executing procedure. Moreover, computational complexity for detecting computer viruses is also discussed briefly.

Keywords Computational complexity, Computer viruses, Injury, Infection, Imitation

1 引言

第一个关于计算机病毒的抽象理论是F. Cohen的基于图灵机的病毒集理论[1,2]。一个病毒集是一个序偶(M,V),其中M是一个图灵机,V是图灵机上一个程序集,每个$v\in V$称为一个计算机病毒,它满足:如果v在时刻t包含于图灵机的带上,则存在另一个时刻t',$v'\in V$,存在于图灵机的带上。基于此,Cohen证明了计算机病毒的不可判定性[1,2]。

和Cohen的途径不同,L. M. Aldeman发展了一个基于递归函数论的计算机病毒理论[3]。在他的定义中,一个计算机病毒是一个递归全函数v,作用于所有程序x上(程序x的Gödel编码),使得$v(x)$具有计算机病毒的典型特征:致损(injury)、传染(infection)和模仿(imitation)。在此基础上,Aldeman证明了更强的计算机病毒不可解定理,即:所有计算机病毒的集合是Π_2完备集[3]。

尽管Cohen和Aldeman的计算机病毒理论及不同的改进[4~6]已经得到详细研究,但对计算机病毒的计算复杂度还知之甚少。Aldeman的文章[3]中提到一些相关问题,但没有给出任何结论。在文[7]中,Spinelis证明存在变体病毒,它的检测问题是完全问题。

本文从数学上证明关于计算机病毒计算复杂度的两个基本结论。第一,存在计算机病毒,它的传染过程具有任意大的计算复杂度;第二,存在计算机病毒,它的执行过程具有任意大的计算复杂度。

本文第2节给出本文用到的一些概念及符号。第3节给出本文采用的计算机病毒的数学定义。第4节给出计算复杂度的定义及相关引理。第5节及第6节分别给出本文结论的数学证明。第7节简要讨论计算机病毒检测过程的计算复杂度问题。

2 概念及符号

令N是自然数集合,S是所有自然数有限序列的集合。对于$s_1,s_2,\cdots,s_n\in S$,$\langle s_1,s_2,\cdots,s_n\rangle$代表一个从$S^n$到$N$的可计算函数,同时具有可计算逆函数。对于函数$f:N\to N$,$s_1,s_2,\cdots,s_n\in S$,$f(s_1,s_2,\cdots,s_n)$表示$f(\langle s_1,s_2,\cdots,s_n\rangle)$。类似,对于$i_1,i_2,\cdots,i_n\in N$,$\langle i_1,i_2,\cdots,i_n\rangle$代表一个从$N^n\to N$的可计算函数,同时具有可计算逆函数,并且$\langle i_1,i_2,\cdots,i_n\rangle\geqslant i_k$,$1\leqslant k\leqslant n$;对于函数$f:N^n\to N$,$f(i_1,i_2,\cdots,i_n)$代表$f(\langle i_1,i_2,\cdots,i_n\rangle)$。符号$f(i_1,i_2,\cdots,i_n)\downarrow$表示$f(i_1,i_2,\cdots,i_2)$有定义,$f(i_1,i_2,\cdots,i_n)\uparrow$表示$f(i_1,i_2,\cdots,i_n)$无定义。

对于序列$p=(i_1,i_2,\cdots,i_k,\cdots,i_n)\in S$,用$(P)_k$代表它的第$k$个元素。$p[j_k/i_k]$代表一个序列,该序列除了用$j_k$代替$i_k$以外,和$p$相同,即$p[j_k/i_k]=(i_1,i_2,\cdots,j_k,\cdots,i_n)$。如果序列$p$中的元素$i_k$被一个函数$v$作用,即$p[v(i_k)/i_k]$,为了符号的简洁性,在文中写为$p[v(\underline{i_k})]$,其中带下划线的符号代表被作用的元素。

一个计算机程序P代表一个从S到S的函数$\varphi_P(d,p)$,它表示程序p在运行环境(d,p)中被计算,其中d和p分别代表数据(包括时钟、磁盘空间等)和程序(包括操作系统)。如

左志宏 副教授,研究方向为计算机网络与信息安全;舒 敏 高级工程师,研究方向为计算机安全;周明天 教授,博导,研究方向为计算机网络与网络安全。

果 P 的 Gödel 编码为 e，该函数写为 $\varphi_e(d,p)$，它的定义域和值域分别写为 W_e 和 E_e。

3 计算机病毒的定义

本文采用的计算机病毒的定义来源于 Aldeman 的定义[3]，但比之更符合通常对计算机病毒的理解[6]。

定义3.1[非驻留型病毒] 满足以下条件的递归全函数 v，称为一个非驻留型计算机病毒：

(1)对所有 $x\in N$，

$$\varphi_{v(x)}(d,p)=\begin{cases} D(d,p), & \text{如果 } T(d,p) \quad \text{(i)}\\ \varphi_x(d,p[v(\underline{S(p)})]), & \text{如果 } I(d,p) \quad \text{(ii)}\\ \varphi_x(d,p) & \text{否则} \quad \text{(iii)}\end{cases} \tag{1}$$

(2)$T(d,p)$ 和 $I(d,p)$ 是两个递归谓词，至少有一个 (d,p) 满足 $I(d,p)$，且没有 (d,p) 同时满足它们；$D(d,p)$ 和 $S(p)$ 是两个递归函数；

(3)集合 $\{\langle d,p\rangle:\neg(T(d,p)\vee I(d,p))\}$ 是无限集。

谓词 $T(d,p)$ 和 $I(d,p)$ 分别称为计算机病毒的致损条件及传染条件；谓词 $T(d,p)$ 被满足，计算机病毒执行致损函数 $D(d,p)$；谓词 $I(d,p)$ 被满足，计算机病毒利用选择函数 $S(p)$ 选择一个可执行程序进行传染，然后执行原程序。两个条件及两个函数合称为计算机病毒的内核，它从数学上唯一决定一个计算机病毒。

定义3.1中方程(1)的三个分支定义了计算机病毒的三个典型行为：致损、传染和模拟。条件(2)保证计算机病毒至少传染一个程序，并且传染和致损不能同时进行。条件(3)是一个强限制条件，它要求一个计算机病毒在无限多个点上模拟原程序。证明具有相同内核的计算机病毒集合是一个 Π_2 完备集合，用到该条件[6]。

按照定义3.1的方式，几乎可以定义所有种类的计算机病毒[6]。下面给出两种重要的计算机病毒的定义，其中不再列出条件(ii)和(iii)，假定它们被满足。

定义3.2[具有两形态的变形病毒] 满足以下条件的两个相异递归全函数对 (v,v')，称为一个具有两形态的变形病毒：对所有的 $x\in N$，

$$\varphi_{v(x)}(d,p)=\begin{cases} D(d,p), & \text{如果 } T(d,p)\\ \varphi_x(d,p[v'(\underline{S(P)})]), & \text{如果 } I(d,p)\\ \varphi_x(d,p), & \text{否则}\end{cases}$$

和

$$\varphi_{v'(x)}(d,p)=\begin{cases} D(d,p), & \text{如果 } T(d,p)\\ \varphi_x(d,p[v(\underline{S(P)})]), & \text{如果 } I(d,p)\\ \varphi_x(d,p), & \text{否则}\end{cases}$$

具有 n 形态的变形病毒定义为满足类似条件的 n 个相异递归函数的序列 $(v_1,v_2,\cdots,v_n)$。变形病毒给检测和清除计算机病毒带来极大麻烦，它们通常具有数十亿种形态，一般来说任意两个形态之间没有连续相同的三个字节。

定义3.3[具有无限形态的变形病毒] 满足以下条件的递归全函数 $v(m,x)$，称为一个具有无限形态的变形病毒：对所有的 $m,x\in N$，

$$\varphi_{v(m,x)}(d,p)=\begin{cases} D(d,p), & \text{如果 } T(d,p)\\ \varphi_x(d,p[v(m+1,\underline{S(p)})]), & \text{如果 } I(d,p)\\ \varphi_x(d,p), & \text{否则}\end{cases}$$

并且对所有 $m\neq n$，$v(m,x)\neq v(n,x)$。

到目前为至，在实际中还未发现具有无限形态的变形病毒。不过它的存在性，和其它病毒一样，理论上可由同一条数学定理证明(递归定理)[6,8,9]。

定义3.4[变体病毒] 满足以下条件的两个相异递归全函数对 (v,v')，称为一个变体病毒[10]：对所有的 $x\in N$，

$$\varphi_{v(x)}(d,p)=\begin{cases} D(d,p), & \text{如果 } T'(d,p)\\ \varphi_x(d,p[v'(\underline{S(P)})]), & \text{如果 } I(d,p)\\ \varphi_x(d,p), & \text{否则}\end{cases}$$

和

$$\varphi_{v'(x)}(d,p)=\begin{cases} D'(d,p), & \text{如果 } T(d,p)\\ \varphi_x(d,p[v(\underline{S'(P)})]), & \text{如果 } I'(d,p)\\ \varphi_x(d,p), & \text{否则}\end{cases}$$

其中，$T(d,p)$(相应的 $I(d,p)$，$D(d,p)$ 和 $S(p)$)与 $T'(d,p)$(相应的 $I'(d,p)$，$D'(d,p)$ 和 $S'(p)$)是两个相异递归谓词(或函数)。

具有 n 变体的变体病毒定义为满足类似条件的相异递归全函数序列 $(v_1,v_2,\cdots,v_n)$。变体病毒与变形病毒的根本差异在于：变形病毒只改变自己的形态，不同形态的病毒具有相同内核，但变体病毒改变内核本身，不同的传染变成不同的病毒(即具有不同的致损条件，不同的致损行为，不同的传染条件，不同的传染行为等)。

4 计算复杂度的定义

任何满足下列条件的函数族 $\{r_e(x)\}_{e\in N}$ 称为资源复杂度函数：

(1)对所有 $e\in N$，$Dom(r_e)=Dom(\varphi_e)$；

(2)$'r_e(x)\square y'$ 是可判定谓词。

其中 Dom 表示函数的定义域。

通常的时间复杂度函数定义如下，满足上面的条件，是一种资源复杂度函数：

$$t_p(d,p)=\begin{cases} P\text{ 计算 }f_p(d,p)\text{ 的步数}, & \text{如果 } f_p(d,p)\text{ 有定义}\\ \uparrow & \text{否则}\end{cases}$$

$$=\mu t(\text{在 } t \text{ 步内 } P(d,p)\downarrow)$$

如果程序 P 的 Gödel 编码为 e，则 t_p 也写作 t_e。空间复杂度函数可类似定义。本文以下讨论计算复杂度时，假定 $\{r_e(x)\}_{e\in N}$ 是相应的资源复杂度函数，不特指何种资源。用符号“$a.e$”表示“除有限个点外满足”，即 $f(x)>g(x)a.e$ 意味着，存在 $n_0\in N$ 使得对任意 $n\geqslant n_0$，$f(n)>g(n)$。在这节中，采用传统递归函数论的概念和符号。

下面给出与资源复杂度有关的两个引理。

引理4.1 令 $b(x)$ 是任一递归全函数，存在递归全函数 f，若 e 是 f 的一个下标，则 $r_e(x)>b(x)a.e$。

证明：见文[8]。

引理4.2 令 $b(x)$ 是任一递归全函数，对任意递归函数 $f(x,y,z)$，存在递归全函数 $k(x,y)$，若 e 是 $k(x,y)$ 的一个下标，则对所有的 y，$r_e(x,y)>b(x)a.e$。

证明：由 $s-m-n$ 定理，存在一个递归全函数 $g(x,y,s)$ 满足

$$\varphi_{g(x,y,s)}(z)=1(s)f(x,y,z)$$

并且对 $s_1\neq s_2$，$g(x,y,s_1)\neq g(x,y,s_2)$。

定义函数

$$k(x,y)=\begin{cases} g(x,y,1), & \text{如果 } i_x \text{ 有定义且 } \varphi_{i_x}(x)=g(x,y,0)\\ g(x,y,0), & \text{否则}\end{cases}$$

其中 i_x 的定义如同引理4.1。

使用和引理4.1中同样的推理，可知 $k(x,y)$ 是一个递归全函数，且若 e 是 $k(x,y)$ 的一个下标，则对所有的 y，$r_e(x,y)>b(x)a.e$。

由于

$$\varphi_{k(x,y)}(z)$$

$$=\begin{cases}\varphi_{g(x,y,1)}(z), & \text{如果 } i_x \text{ 有定义且 } \varphi_x(x)=g(x,y,0)\\ \varphi_{g(x,y,0)}(z), & \text{否则}\end{cases}$$

$$=\begin{cases}1(1)f(x,y,z), & \text{如果 } i_x \text{ 有定义且 } \varphi_x(x)=g(x,y,0)\\ 1(0)f(x,y,z), & \text{否则}\end{cases}$$

$$=f(x,y,z),$$

故 $k(x,y)$满足引理要求。证毕。

引理4.2可看作是对传统的 $s-m-n$ 的定理一个改进，它表明下标函数可以是任何递归函数，而不仅仅是 $s-m-n$ 定理中的原始递归函数。引理4.2可以推广到包含任何的参数。

引理4.3 对任意 $m_1,m_2,n\geqslant 1$，令 $n=m_1+m_2$，$b(x)$是任一 m_1元递归全函数。存在 $m+1$元递归全函数 $s_n^m(c,x,y)$，使得 $\varphi_x^{m+n}(x,y,z)=\varphi_{s_n^m(c,x,y)}^n(z)$并且若 e 是 $s_n^m(c,x,y)$的下标，则对所有 $y,r_e(x,y)>b(x)a.e.$

证明：和引理4.2类似，略。

5 计算机病毒传染过程的计算复杂度

利用引理4.3，可直接证明存在任意类型的计算机病毒 v，它的传染过程具有任意大的计算复杂度，即它在传染过程中可以消耗任意多的计算机资源。

令 $b(x)$是任意递归全函数，考虑如下函数：

$$f(k,x,\langle d,p\rangle)=\begin{cases}D(d,p), & \text{如果 } T(d,p)\\ \varphi_x(d,p[\varphi_k(\underline{S(p)})]) & \text{如果 } I(d,p)\\ \varphi_x(d,p), & \text{否则}\end{cases}$$

由引理4.3可知，存在递归全函数 $h(k,x)$，若 e 是它的一个下标，则对所有的 $y,,r_e(x,y)>b(x)a.e.$，同时，

$$\varphi_{h(k,x)}(d,p)=f(k,x,\langle d,p\rangle)$$

$$=\begin{cases}D(d,p) & ,\text{如果 } T(d,p)\\ \varphi_x(d,p[\varphi_k(\underline{S(p)})]), & \text{如果 } I(d,p)\\ \varphi_x(d,p), & \text{否则}\end{cases}$$

由 $s-m-n$ 定理，存在递归全函数 $g(k)$，$\varphi_{g(k)}(x)=h(k,x)$。由递归定理，存在一个 n，$\varphi_{g(n)}=\varphi_n$。令 $v(x)=h(n,x)=\varphi_{g(n)}(x)=\varphi_n(x)$，则

$$\varphi_{v(x)}(d,p)=\varphi_{h(n,x)}(d,p)=f(n,x,\langle d,p\rangle)$$

$$=\begin{cases}D(d,p), & \text{如果 } T(d,p)\\ \varphi_x(d,p[\varphi_n(\underline{S(p)}], & \text{如果 } I(d,p)\\ \varphi_x(d,p), & \text{否则}\end{cases}$$

$$=\begin{cases}D(d,p), & \text{如果 } T(d,p)\\ \varphi_x(d,p[v(\underline{S(p)}]), & \text{如果 } I(d,p)\\ \varphi_x(d,p), & \text{否则}\end{cases}$$

由定义3.1，上式中的 v 是一个非驻留计算机病毒。同时，若 e 是它的一个下标，则 $r_e(x)>b(x)a.e.$该证明方法可以应用到所有其他类型的计算机病毒，因此有如下定理。

定理5.1 对任意递归全函数 $b(x)$，存在任意类型的计算机病毒 v，若 e 是它的一个下标，则 $r_e(x)>b(x)a.e.$

6 计算机病毒执行过程的计算复杂度

定理6.1 对任意递归全函数 $b(x)$，存在计算机病毒 v，若 φ_x 是全函数，则 $r_{v(x)}(d,p)>b(d,p)a.e.$

证明：对于任意 $k,x\in N$，考虑如下函数：

$$f(k,x,\langle d,p\rangle)=$$

$$\begin{cases}1, & \text{如果 } i_{(d,p)} \text{有定义且 } \varphi_{(d,p)}(d,p)=0\\ 0 & \text{如果 } i_{(d,p)} \text{有定义且 } \varphi_{(d,p)}(d,p)\neq 0\\ \varphi_x(d,p[\varphi_k((p)_1)]), & \text{如果 } i_{(d,p)} \text{有定义且} (d)_1=0\\ \varphi_x(d,p) & \text{否则}\end{cases}$$

其中 $i_{(d,p)}$的定义类似于引理4.1中的 i_x(用$\langle d,p\rangle$代替 x)。使用和引理4.1同样的推理可得，如果 φ_x 是全函数，则 $f(k,x,\langle d,p\rangle)$是递归全函数；若 e 是 $f(k,x,\langle d,p\rangle)$的下标，则 $r_e(d,p)>b(d,p)a.e.$

使用与定理5.1相同的推理，可知存在一个递归全函数 $v(x)$，满足：

$$\varphi_{v(x)}(d,p)=$$

$$\begin{cases}1, & \text{如果 } i_{(d,p)} \text{有定义且 } \varphi_{(d,p)}(d,p)=0(i)\\ 0 & \text{如果 } i_{(d,p)} \text{有定义且 } \varphi_{(d,p)}(d,p)\neq 0(ii)\\ \varphi_x(d,p[v((p)_1)]), & \text{如果 } i_{(d,p)} \text{有定义且} (d)_1=0(iii)\\ \varphi_x(d,p) & \text{否则}\end{cases}$$

由定义3.1可知，上式中的 v 是一个非驻留计算机病毒(其中，分支(i)和(ii)是它的致损行为；分支(iii)是它的传染行为；分支是它的模拟行为)；同时，$v(x)$是函数 $f(k,x,\langle d,p\rangle)$的一个下标，所以 $r_{v(x)}(d,p)>b(d,p)a.e.$ 证毕。

对于每一个 $x\in N$，$v(x)$是一个被病毒 v 感染的程序，因此 $r_{v(x)}(d,p)>b(d,p)a.e$ 意味着该程序在几乎所有(d,p)(除了有限多个外)上的执行代价大于 $b(d,p)$。由于 $b(d,p)$是任意指定的递归全函数，故 $v(x)$的执行代价可以任意大。

7 相关问题

与计算机病毒的计算复杂度相关的另一个重要问题是计算机病毒检测过程的计算复杂度问题。对任意计算机病毒 v，令 $I_v=E_v=\{v(x):x\in N\}$[3]，计算机病毒检测的计算复杂度问题就是该集合的计算复杂度问题。关于计算机病毒的一个重要结论是，存在计算机病毒 v，使得 I_v 是不可判定集合[1~3]。因此计算机病毒检测的计算复杂度问题可具体化为：

(1)如果 I_v 是可判定的，I_v 的计算复杂度如何？是否存在计算机病毒，使得 I_v 具有任意大的计算复杂度？

(2)如果 I_v 是不可判定的，对于任意满足 $I_v\subset C$ 的递归集 C，C 具有怎样的计算复杂度？

第一个问题的回答是肯定的，即存在计算机病毒 v，使得 I_v 具有任意大的计算复杂度。由于篇幅所限，未给出证明。第二个问题只是部分解决，即可以证明存在递归集 C，$I_v\subset C$ 且 C 具有任意大的计算复杂度。但 C 的其它计算复杂度特征(比如，对任意病毒 v，是否存在非平凡递归集 C，$I_v\subset C$ 且 C 具有线性时间复杂度？)，还在进一步的研究当中。

参考文献

1 Cohen F. Computational aspects of computer viruses. Computers & Security, 1989,8(4):325～344

2 Cohen F. A Short Course on Computer Viruses. Wiley,1994

3 Adleman L M. An abtract theory of computer viruses. In: Advances in Cryptology--CRYPTO'88, LNCS,403, Goldwasser, S, ed. Berlin: Springer-Verlag,1990. 354～374

4 王剑，唐朝京，张权，等. 基于扩展通用图灵机的计算机病毒传染模型. 计算机研究与发展，2003,40(9):1300～1306

5 田畅，郑少仁. 计算机病毒计算模型的研究. 计算机学报，2001,24(2):158～163

6 Zuo Zhihong, Zhou Mingtian. Some further theoretical results about computer viruses. The Computer Journal,2004,47(6)

7 Spinellis D. Reliable identification of bounded-length viruses is NP-complete. IEEE transactions on information theory, 2003, 49 (1):280～284

8 Cutland N. Computability: introduction of recursive function theory. Cambridge University Press,1980

9 Rogers H Jr. Theory of Recursive Functions and Effective Computability. McGraw-Hill,1967

10 Szor P, Ferrie P. Hunting for metamorphic. online http://www.virusbtn.com,2000