available at www.sciencedirect.com
ScienceDirect
journal homepage: www.elsevier.com/locate/cose
Computers & Security

Infection, imitation and a hierarchy of computer viruses

Zhi-hong Zuo, Qing-xin Zhu, Ming-tian Zhou*

College of Computer Science and Engineering, University of Electronic Science & Technology of China, Chengdu, Sichuan 610054, PR China

ARTICLE INFO

Article history:
Received 19 May 2005
Revised 12 October 2005
Accepted 6 February 2006

Keywords:
Computer viruses
Infection
Imitation
Complete sets
Hierarchy

ABSTRACT

Infection is an essential character of computer viruses. In addition, computer viruses can also imitate the behavior of infected programs in some ways in order to hide themselves. In this paper we define infection and imitation mathematically, and classify computer viruses into 3 types according to their different imitation behaviors. Furthermore, we give some results about the degree of unsolvability of each type of computer viruses. We show that the set of type 0 and type 1 computer viruses is Π_2-complete, while the set of type 2 computer viruses is Π_3-complete.

1. Introduction

The first abstract theory of computer viruses is the viral set theory given by Cohen, based on the Turing machine (Cohen, 1989, 1994). A viral set is defined by (M, V) where M is a Turing machine and V is a non-empty set of programs on M. Each $v \in V$ is called a computer virus and satisfies the following conditions: if it is contained in the tape at time t, then there is a time t' and a $v' \in V$ such that v' is not contained in the tape at time t, but contained in the tape at time t'. The most important one of Cohen's theorems is about the undecidability of computer viruses (Cohen, 1989, 1994).

In a different approach, Adleman (1988) developed an abstract theory of computer viruses based on recursive functions. In his definition a virus is a total recursive function v which applies to all programs x (the Gödel numberings of programs) such that $v(x)$ has characteristic behaviors of computer viruses such as *injury*, *infection* and *imitation*. Furthermore, Adleman (1988) proved that the set of computer viruses is Π_2-complete.

There are some shortcomings in the computer virus models given by Cohen and Adleman. Several improvements have been proposed so far (Thimbleby et al., 1999; Jian et al., 2003; Chang and Shao-Ren, 2001; Zuo and Zhou, 2004). In a recent paper (Zuo and Zhou, 2004), we improved Adleman's definitions of computer viruses to comply with the common understanding of computer virus, and proved that the set of computer viruses with the same kernel is Π_2-complete. In general they formed a Σ_3-complete set. We have also proved theoretically the existence of some computer viruses that have not been discovered yet (for example, the polymorphic viruses with infinite forms). In another paper (Zhi-hong et al., 2005), we investigated the time complexity of computer viruses.

Infection is the key character of computer viruses. In addition, computer viruses often imitate the behavior of the infected programs in some ways in order to hide themselves. Different imitation behaviors lead to different mathematical features. In this paper we define infection and imitations mathematically, obtain a hierarchical structure of computer viruses according to their different imitation behaviors and prove the strict inclusions.

The structure of this paper is as follows: in Section 2 we introduce some basic concepts and notations; in Section 3 we give the definitions of infection, imitation and a hierarchical

* Corresponding author.
E-mail address: zhzuo@uestc.edu.cn (Z.-hong Zuo).
0167-4048/$ – see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2006.02.001

structure of computer viruses. In Section 4 we give some important theorems and prove them. In Section 5, we give a brief summary and some discussion for these results.

2. Preliminaries

We describe some notations below.

Let $\mathbb{N}$ be the set of all natural numbers and S be the set of all finite sequences of natural numbers. For $s_1, s_2, \ldots, s_n \in S$, let $\langle s_1, s_2, \ldots, s_n \rangle$ denote a computable injective function from S^n to $\mathbb{N}$ and its inverse is also computable. If $f : \mathbb{N} \to \mathbb{N}$ is a partial function, for $s_1, s_2, \ldots, s_n \in S$, we write $f(s_1, s_2, \ldots, s_n)$ instead of $f(\langle s_1, s_2, \ldots, s_n \rangle)$. Similarly, for $i_1, i_2, \ldots, i_n \in \mathbb{N}$, let $\langle i_1, i_2, \ldots, i_n \rangle$ denote a computable injective function from $\mathbb{N}^n$ to $\mathbb{N}$, satisfying $\langle i_1, i_2, \ldots, i_n \rangle \geq i_m$ for all $1 \leq m \leq n$, and its inverse is also computable. We also use $f(i_1, i_2, \ldots, i_n)$ to represent $f(\langle i_1, i_2, \ldots, i_n \rangle)$.

For a sequence $p = (i_1, i_2, \ldots, i_k, \ldots, i_n) \in S$, let $p(i)$ denote its ith element, and $x \in_s p$ represent that x is in the sequence p, i.e., $x = p(i)$ for some i. For $s_1, s_2, \ldots, s_n \in S$, $x \in_s (s_1, s_2, \ldots, s_n)$ means $x \in_s s_i$ for some $1 \leq i \leq n$. Let $p[j_k/i_k]$ denote the sequence obtained by replacing i_k with j_k in p, i.e., $p[j_k/i_k] = (i_1, i_2, \ldots, j_k, \ldots, i_n)$. If v is a computable function, $p[v(i_k)/i_k]$ is simply written as $p[v(\underline{i_k})]$. If more than one element in p is replaced or evaluated by some computable functions, we write the result as $p[j_{k_1}/i_{k_1}, j_{k_2}/i_{k_2}, \ldots, j_{k_l}/i_{k_l}]$ or $p[v_1(\underline{i_{k_1}}), v_2(\underline{i_{k_2}}), \ldots, v_l(\underline{i_{k_l}})]$, respectively.

Adopting Adleman's (1988) notations, let $\phi_P(d, p)$ denote a function computed by a computer program P in the running environment (d, p) where d represents *data* (including clock, spaces of diskettes and so on) and p represents *programs* (including operating systems) stored on computers. If the index (the Gödel numbering) of P is e, the function is also denoted by $\phi_e(d, p)$. The domain and range are denoted by W_e and E_e, respectively. If h is a recursive function, we also use the symbols W_h and E_h for its domain and range. It is worth noting that there is no essential distinction between d and p, as in the case of Von Neumann machines. In this paper we use the symbol (d, p) just for easier understanding.

3. Definitions of infection, imitation, and the hierarchy of computer viruses

In the following we give definitions of infection and imitation first, and then derive the hierarchy structure for computer viruses.

A computer virus can be viewed as a total recursive function v which applies to every program i and obtains its infected form $v(i)$ such that $v(i)$ can infect other programs (or MBR as well as some documentations) under some conditions (Adleman, 1988). In more technical terms, an infected program $v(i)$, when given some input (or environments) (d, p), its output $\phi_{v(i)}(d, p)$ should contain some other infected programs $v(x)$. It leads to the following definition of infection.

Definition 1. (*Infection*) A total recursive function v is said to be infective if it satisfies

$$\forall i\Big(W_i \neq \emptyset \to \exists \langle d, p \rangle \in W_{v(i)}\Big(\exists x\Big(v(x) \in_s \phi_{v(i)}(d, p)\Big)\Big)\Big). \tag{1}$$

Imitation is a property upon which computer viruses rely to behave like the original programs. It is not indispensable for computer viruses, but most currently found computer viruses do have imitation property.

Definition 2. (*Imitation*) A total recursive function v is said to be imitative if it satisfies

$$\forall i\Big(|W_i| > 2 \to \exists \langle d, p \rangle \in W_i \cap W_{v(i)}\Big(\phi_{v(i)}(d, p) = \phi_i(d, p)\Big)\Big). \tag{2}$$

Imitation property makes the infected program $v(i)$ behave in some computations like the original program i, i.e., there exist some environments (d, p), $\phi_{v(i)}(d, p) = \phi_i(d, p)$.

Definition 3. (∞*-Imitation*) A total recursive function v is said to be ∞-imitative if it satisfies

$$\forall i\Big(W_i \text{ is infinite} \to \exists^\infty \langle d, p \rangle \in W_i \cap W_{v(i)}\Big(\phi_{v(i)}(d, p) = \phi_i(d, p)\Big)\Big), \tag{3}$$

where symbol $\exists^\infty$ means existing infinitely many.

∞-Imitation property not only requires the infected program $v(i)$ behave in some computations like the original program i, but also requires infinitely many of these computations.

Definition 4. (*Computer virus*) A computer virus is a total a recursive function v satisfying the following conditions:

(1) v has infection property, or
(2) v has both infection and imitation property, or
(3) v has both infection and ∞-imitation property.

Computer viruses satisfying the above conditions (1), (2) or (3) are called type 0, type 1 or type 2 viruses, respectively. The set of type 0, type 1 and type 2 viruses are denoted by V_0, V_1 and V_2, respectively. Obviously $V_0 \supseteq V_1 \supseteq V_2$.

4. Main results

In this section we prove our main results, using the traditional notations and symbols of recursive function theory (Rogers, 1967; Soare, 1987).

Proposition 5. *"v is infective" is a Π_2-predicate.*

Proof. From Definition 1, it follows that
"v is infective"

$$\Leftrightarrow \forall i\Big(W_i \neq \emptyset \to \exists \langle d, p \rangle \in W_{v(i)}\Big(\exists x\Big(v(x) \in_s \phi_{v(i)}(d, p)\Big)\Big)\Big) \tag{4}$$

$$\Leftrightarrow \forall i\Big((W_i = \emptyset) \vee \Big(\exists \langle d, p \rangle \in W_{v(i)}\Big(\exists x\Big(v(x) \in_s \phi_{v(i)}(d, p)\Big)\Big)\Big)\Big). \tag{5}$$

Since

$$W_i = \emptyset \Leftrightarrow \forall x. x \notin W_i \tag{6}$$

$$x \in_s \phi_y(z) \Leftrightarrow \exists i\Big(x = \phi_y(z)(i)\Big), \tag{7}$$

we know that "$W_i = \emptyset$" and "$x \in_s \phi_y(z)$" are a Π_1-predicate and a Σ_1-predicate, respectively. Because "$x \in W_y$" is also a Σ_1-predicate (Rogers, 1967), "υ is infective" is a Π_2-predicate. □

Proposition 6. *"υ is imitative" is a Π_2-predicate.*

Proof. From Definition 2, it follows that

"υ is imitative"

$$\Leftrightarrow \forall i\Big(|W_i| > 2 \rightarrow \exists \langle d,p\rangle \in W_i \cap W_{\upsilon(i)}\Big(\phi_{\upsilon(i)}(d,p) = \phi_i(d,p)\Big)\Big) \tag{8}$$

$$\Leftrightarrow \forall i\Big((|W_i| \le 2) \vee \Big(\exists \langle d,p\rangle \in W_i \cap W_{\upsilon(i)}\Big(\phi_{\upsilon(i)}(d,p) = \phi_i(d,p)\Big)\Big)\Big). \tag{9}$$

Since

$$|W_i| > 2 \Leftrightarrow \exists x,y,z \in W_i(x \ne y \wedge y \ne z \wedge x \ne z), \tag{10}$$

we know that "$|W_i| > 2$" is a Σ_1-predicate, hence "$|W_i| \le 2$" is a Π_1-predicate. Since "$x \in W_y$" is *a* Σ_1-predicate, "υ is imitative" is a Π_2-predicate. □

Proposition 7. *"υ is ∞-imitative" is a Π_3-predicate.*

Proof. From Definition 3, it follows that

"υ is ∞-imitative"

$$\Leftrightarrow \forall i\Big(W_i \text{ is infinite} \rightarrow \exists^\infty \langle d,p\rangle \in W_i \cap W_{\upsilon(i)}\Big(\phi_{\upsilon(i)}(d,p) = \phi_i(d,p)\Big)\Big) \tag{11}$$

$$\Leftrightarrow \forall i\Big((W_i \text{ is finite}) \vee \Big(\forall x \exists \langle d,p\rangle \in W_i \cap W_{\upsilon(i)}\Big((\langle d,p\rangle > x) \wedge \Big(\phi_{\upsilon(i)}(d,p) = \phi_i(d,p)\Big)\Big)\Big)\Big). \tag{12}$$

Since "W_i is finite" is a Σ_2-predicate (Rogers, 1967), and "$x \in W_y$" is a Σ_1-predicate, "υ is ∞-imitative" is a Π_3-predicate. □

In the proofs of our main results, we also need the following lemma.

Lemma 8. *For any non-empty recursively enumerable set R, there is a recursive function $\Psi(e, y)$, such that $Rng(\lambda y.\Psi(e, y)) = W_e \bigcup R$ for any e. The set is also written as E_e^R.*

Proof. Let $R = Rng(g)$, here g is a recursive function. Define

$$f(e,x,n) = \begin{cases} g(s), & \text{if } n = 2s+1 \\ x, & \text{if } n = 2s, x \in W_{e,s+2} - W_{e,s} \\ g(0), & \text{otherwise} \end{cases} \tag{13}$$

where $W_{e,s}$ is defined as in Soare (1987). Clearly $f(e, x, n)$ is a total recursive function. Let $\Psi(e, y) = f(e, l(y), r(y))$, we have the conclusion. □

Theorem 9. *V_0 and V_1 are Π_2-complete sets.*

Proof. Since

$$\upsilon \in V_0 \Leftrightarrow \upsilon \text{ is infective} \tag{14}$$

$$\upsilon \in V_1 \Leftrightarrow (\upsilon \in V_0) \wedge (\upsilon \text{ is imitative}), \tag{15}$$

by Propositions 6 and 7, we know that V_0 and V_1 are Π_2-sets.

Let A be any Π_2-set, $R(x, y, z)$ be the recursive predicate satisfying $x \in A \Leftrightarrow \forall y \exists z R(x,y,z)$. Let a be an integer, assume $R = \{a\}$ and by Lemma 8 we have $\Psi(e,y)$. Let $b = \Psi(i, \mu y(\Psi(i, y) \ne a))$. Clearly b is a recursive function of i. For a given number m, define

$$f(i,k,x,\langle d,p\rangle) = \begin{cases} \langle d,p,\phi_k(m)\rangle, & \text{if}((\langle d,p\rangle = a) \vee (\langle d,p\rangle = b)) \\ & \wedge(\forall y < \langle d,p\rangle \exists z R(x,y,z)) \\ \phi_i(d,p), & \text{if}(\langle d,p\rangle \in E_i^a) \wedge (\forall y < \langle d,p\rangle \exists z R(x,y,z)) \\ \text{undefined}, & \text{otherwise} \end{cases} \tag{16}$$

$f(i, k, x, \langle d, p\rangle)$ can be computed by the following procedure.

Given $(i, k, x, \langle d, p\rangle)$, compute $\Psi(i, 0), \Psi(i, 1), \ldots$ starting from 0. Let $\langle d, p\rangle = \Psi(i, j)$, when a value of $\langle d, p\rangle$ is computed, we compute the sequence $R(x,0,0), \ldots, R(x,\langle d,p\rangle,0), R\ldots(x,0,1), \ldots, R(x,\langle d,p\rangle,1), \ldots$. If for every $y < \langle d, p\rangle$ there is a z such that the value of $R(x, y, z)$ is 1 (true), then check if $\langle d, p\rangle$ is equal to a or b (provided b exists); if equal, the procedure outputs $\langle d, p, \phi_k(m)\rangle$; otherwise outputs $\phi_i(d, p)$; in other situations (including the case where b does not exist), the procedure does not terminate, that is, $f(i, k, x, \langle d, p\rangle)$ is undefined. By Church's thesis, $f(i, k, x, \langle d, p\rangle)$ is a recursive function.

By *s–m–n* theorem, there exists a total recursive function $b(i, j, k)$ satisfying

$$\phi_{b(i,k,x)}(d,p) = \begin{cases} \langle d,p,\phi_k(m)\rangle, & \text{if}((\langle d,p\rangle = a) \vee (\langle d,p\rangle = b)) \\ & \wedge(\forall y < \langle d,p\rangle \exists z R(x,y,z)) \\ \phi_i(d,p), & \text{if}(\langle d,p\rangle \in E_i^a) \wedge (\forall y < \langle d,p\rangle \exists z R(x,y,z)) \\ \text{undefined}, & \text{otherwise} \end{cases} \tag{17}$$

By the recursion theorem with parameters, there exists a total recursive function $n(x)$ such that $\phi_{n(x)}(i) = b(i, n(x), x)$, hence

$$\phi_{\phi_{n(x)}(i)}(d,p) = \begin{cases} \langle d,p,\phi_{n(x)}(m)\rangle, & \text{if}((\langle d,p\rangle = a) \vee (\langle d,p\rangle = b)) \\ & \wedge(\forall y < \langle d,p\rangle \exists z R(x,y,z)) \\ \phi_i(d,p), & \text{if}(\langle d,p\rangle \in E_i^a) \wedge (\forall y < \langle d,p\rangle \exists z R(x,y,z)) \\ \text{undefined}, & \text{otherwise} \end{cases} \tag{18}$$

If $x \in A$, then $\forall y \exists z R(x, y, z)$, therefore

$$\phi_{\phi_{n(x)}(i)}(d,p) = \begin{cases} \langle d,p,\phi_{n(x)}(m)\rangle, & \text{if}((\langle d,p\rangle = a) \vee (\langle d,p\rangle = b) \\ \phi_i(d,p), & \text{if}\langle d,p\rangle \in E_i^a \\ \text{undefined}, & \text{otherwise} \end{cases} \tag{19}$$

For any i, if $a \in W_i$ or $b \in W_i$, in both cases $\phi_{n(x)}(m) \in_s \phi_{\phi_{n(x)}(i)}(d,p)$, i.e., $\phi_{n(x)}$ is infective. Moreover, assume $|W_i| > 2$, then $|W_i - \{a, \ b\}| > 0$. From Eq. (19), we have $\phi_{\phi_{n(x)}(i)}(d,p) = \phi_i(d,p)$ for any $\langle d, p\rangle \in W_i - \{a, b\}$, i.e., $\phi_{n(x)}$ is imitative. Hence, for any $x \in A$, $n(x) \in V_1$.

If $x \notin A$, then $\exists y \forall z \neg R(x, y, z)$. Let y_0 be an integer such that $\forall z \neg R(x, y_0, z)$, and let $W_c = \{\langle d, p\rangle : \langle d, p\rangle > y_0\}$, for any $\langle d, p\rangle \in W_c$, we have that

$$y_0 < \langle d,p\rangle \Rightarrow \exists y < \langle d,p\rangle \forall z \neg R(x,y,z). \tag{20}$$

Thus $\phi_{\phi_{n(x)}(c)}(d,p)$ is undefined, that is, for any m, $\phi_{n(x)}(m) \notin \phi_{\phi_{n(x)}(c)}(d,p)$. Hence $n(x)$ is not infective, so that $n(x) \notin V_0$.

In conclusion, $A \le_m (V_1, \overline{V}_0)$. Hence V_0 and V_1 are Π_2-complete sets. This completes the proof of the theorem. □

Theorem 10. *V_2 is a Π_2-complete set.*

Proof. Since $v \in V_2 \Leftrightarrow v \in V_0 \wedge$ "v is ∞-imitative", by Theorem 9 and Proposition 7 V_2 is a Π_3-set.

Let A be any Π_3-set, hence there is a Σ_2-predicate $R(x, y)$ such that $x \in A \Leftrightarrow \forall y R(x, y)$. Since the set $\{x: W_x$ is finite$\}$ is Σ_2-complete, there is a recursive function $g(x, y)$ satisfying $x \in A \Leftrightarrow \forall y(W_{g(x,y)}$ is finite).

For any integer a, let $R = \{a\}$ in Lemma 8 we get the function $\Psi(e, y)$. For a given i, let

$$b_1 = \Psi(i, \mu y(\Psi(i, y) \neq a)), \tag{21}$$

$$b_2 = \Psi(i, \mu y((\Psi(i, y) \neq a) \wedge (\Psi(i, y) \neq b))). \tag{22}$$

It is obvious that b_1 and b_2 are recursive functions with respect to i, and $a \neq b_1 \neq b_2$.

Given m, define

$$f(i, k, x, \langle d, p \rangle) = \begin{cases} \langle d, p, \phi_k(m) \rangle, & \text{if}(\langle d, p \rangle = a) \vee (\langle d, p \rangle = b_1) \\ \phi_i(d, p), & \text{if}\langle d, p \rangle = b_2 \\ \phi_i(d, p), & \text{if}\forall y \leq i(\Psi(g(x, y), \langle d, p \rangle) = a) \\ \text{undefined}, & \text{otherwise} \end{cases} \tag{23}$$

$f(i, k, x, \langle d, p \rangle)$ can be computed by the following procedure.

Given $(i, k, x, \langle d, p \rangle)$, first compute b_1 and b_2. If $\langle d, p \rangle$ equals a or b_1 (provided b_1 exists), then compute $\langle d, p, \phi_k(m) \rangle$; if $\langle d, p \rangle$ equals b_2 (provided b_2 exists), then compute $\phi_i(d, p)$; otherwise compute $\Psi(g(x, 0), \langle d, p \rangle)$, $\Psi(g(x, 1), \langle d, p \rangle)$, …, $\Psi(g(x, i), \langle d, p \rangle)$, if all the values of the sequence are equal to a, then compute $\phi_i(d, p)$; for any other cases (including the case when b_1 and b_2 do not exist), $f(i, k, x, \langle d, p \rangle)$ are undefined. By Church's thesis, $f(i, k, x, \langle d, p \rangle)$ is a recursive function.

Similar to Theorem 9, there exists a recursive function $n(x)$ satisfying

$$\phi_{\phi_{n(x)}(i)}(d, p) = \begin{cases} \langle d, p, \phi_{n(x)}(m) \rangle, & \text{if}(\langle d, p \rangle = a) \vee (\langle d, p \rangle = b_1) \\ \phi_i(d, p), & \text{if}\langle d, p \rangle = b_2 \\ \phi_i(d, p), & \text{if}\forall y \leq i(\Psi(g(x, y), \langle d, p \rangle) = a) \\ \uparrow, & \text{otherwise} \end{cases} \tag{24}$$

and $n(x)$ is both infective and imitative.

If $x \in A$, then for any i, $W_{g(x,i)}$ is finite. By the definition of $\Psi(e, y)$, the function $\lambda z.\Psi(g(x, i), z)$ does not equal a for finitely many z. Hence for all $y \leq i$, the function $\lambda z.\Psi(g(x, i), z)$ does not equal a only for finitely many z. Therefore if W_i is an infinite set, there are infinitely many $\langle d, p \rangle \in W_i$ satisfying $\forall y \leq i(\Phi(g(x, y), \langle d, p \rangle) = a)$, i.e., $\phi_{\phi_{n(x)}(i)}(d, p) = \phi_i(d, p)$. So that $n(x) \in V_2$.

If $x \notin A$, there exists a y' such that $W_{g(x,y)}$ is an infinite set. Let $T = \{\langle d, p \rangle: \exists y < y'(\Psi(g(x, y), \langle d, p \rangle) \neq a)\}$. Clearly T is an infinite recursively enumerable set. Let $c > y'$ and $W_c = T$, $\langle d, p \rangle \in W_c$ and does not equals b_2. If $\phi_{\phi_{n(x)}(c)}(d, p) = \phi_c(d, p)$, from Eq. (24) we have

$$\forall y < c(\Psi(g(x, y), \langle d, p \rangle) = a) \Rightarrow \forall y < y'(\Psi(g(x, y), \langle d, p \rangle) = a). \tag{25}$$

Meanwhile, by the definition of set T, we have

$$\langle d, p \rangle \in T \Leftrightarrow \exists y \leq y'(\Psi(g(x, y), \langle d, p \rangle) \neq a). \tag{26}$$

That lead to a contradiction. Therefore, though W_c is an infinite set, only $b_2 \in W_c$ can satisfy $\phi_{\phi_{n(x)}(c)}(b_2) = \phi_c(b_2)$. Hence we have $n(x) \notin V_2$ for $x \notin A$.

In conclusion, $A \leq_m V_2$, so V_2 is a Π_3-complete set. □

Because V_2 is a Π_3-complete set while V_1 is a Π_2-complete set, hence $V_1 \neq V_2$. Given m, define a function v such that (by recursion theorem)

$$\phi_{v(i)}(d, p) = \langle d, p, v(m) \rangle. \tag{27}$$

Clearly $v \in V_0$ but $v \notin V_1$, hence we have the following theorem.

Theorem 11. $V_0 \supset V_1 \supset V_2$

5. Discussion

Definition 1 is a reasonable description for infective property of computer viruses. But it is not the strongest definition. The strongest definition implies that no matter whether W_i is empty or not, program $\phi_{v(i)}$ is infective. That is,

$$\forall i \exists \langle d, p \rangle \in W_{v(i)} \left(\exists x \left(v(x) \in_s \phi_{v(i)}(d, p) \right) \right). \tag{28}$$

Under such condition we can prove that V_1 is Π_2-complete and V_2 is Π_3-complete, using the same arguments as in Theorems 9 and 10. But to prove that whether V_0 is Π_2-complete or not is still an open problem.

The condition "W_i is not empty" in Definition 1 is replaced by the condition "$|W_i| > 2$" in Definition 2. If we only consider imitative property, we may use the condition "W_i is not empty". If we consider infective property together with imitative property, this is not a proper condition for if $|W_i| = 1$ the program $\phi_{v(i)}$ cannot satisfy both infective and imitative property. Although "$|W_i| > 1$" is the best substitute for the condition "W_i is not empty", we do not know if V_1 is Π_2-complete under this condition.

The conclusion of Theorem 9 complies to some extent with Adleman's result on computer viruses (Adleman, 1988). That is, the decision problems for type 0 and type 1 computer viruses are unsolvable, and the degree of unsolvability is 2 (that is, solving these problems are harder than solving halting problem). Theorem 10 shows that the decision problem of type 2 viruses is even harder than that of type 0 and type 1 computer viruses. Most computer viruses currently found are type 2 viruses. They are both infective and imitative, imitating infected programs in infinitely many computations (or environments).

In the definition of computer viruses (Definition 4), infective property is a necessary condition for a computer virus. Some illegal programs (malicious programs) which do not have infective property are also called computer viruses in a less strict sense. These situations are not included in that definition.

Acknowledgements

The authors wish to thank the referees for their helpful comments that improved the article greatly.

REFERENCES

Adleman LM. An abstract theory of computer viruses. In: Goldwasser S, editor. Advances in cryptology – CRYPTO'88.

Lecture notes in computer science, vol. 403. Berlin: Springer-Verlag; 1988. p. 354–74.
Chang T, Shao-Ren Z. Computational model of computer virus. Chinese Journal of Computers 2001;24(2):158–63.
Cohen F. Computational aspects of computer viruses. Computers & Security 1989;8(1):325–44.
Cohen F. A short course on computer viruses. Wiley; 1994.
Jian W, Chao-Jing T, Quan Z. A computer viruses' infection model based on an expanded universal Turing machine. Journal of Computer Research and Development 2003;40(9): 1300–6.
Rogers HJ. Theory of recursive functions and effective computability. McGraw-Hill; 1967.
Soare RI. Recursively enumerable sets and degrees. Springer-Verlag; 1987.
Thimbleby H, Anderson S, Cairns P. A framework for modelling trojans and computer virus infection. Computer Journal 1999; 41:444–58.
Zhi-hong Z, Qing-xing Z, Ming-tian Z. On the time complexity of computer viruses. IEEE Transaction on Information Theory 2005;51(8):2962–6.
Zuo Z, Zhou M. Some further theoretical results about computer viruses. Computer Journal 2004;47(6):625–33.

Zhi-hong Zuo received the M. Eng. degree in computer software and a Ph.D. degree in computer science both from College of Computer Science and Engineering, University of Electronic Science and Technology of China. Currently he is an associate professor in the college and has published more than 20 journal papers and conference presentations. His research interests are in information security, semantic web and natural language processing.

Qing-xin Zhu received B. Sc. degree from Sichuan Normal University and M. Sc. degree from Beijing Institute of Technology, China. He received M. Sc. degree from Carleton University, Ottawa, ON, Canada, and Ph.D. degree from Ottawa University, Ottawa, ON, Canada. He became a faculty member of the Univeristy of Electronic Science and Technology of China (UESTC) in 1998. He is now a professor of College of Computer Science and Engineering, UESTC. His research interests include network security, computer vision, optimal search and optimal control, and bioinfomatics.

Ming-tian Zhou received E.E.B.S. degree from Harbin Institute of Technology, Harbin, China in 1962. He became a faculty member at the University of Electronic Science and Technology of China (UESTC) in 1962. He is now a professor of College of Computer Science and Engineering, UESTC. He has published 13 books and more than 200 papers. His research interests include network computing, computer network, middleware technology, and network and information security. Prof. Zhou is a fellow of China Institute of Electronics, and a Senior Member of Federation of China Computer.