Some Further Theoretical Results about Computer Viruses

ZHIHONG ZUO AND MINGTIAN ZHOU

School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, P.R. China
Email: zhzuo@263.net

In this paper we give some general definitions of computer viruses which comply with our common understanding of computer viruses. Based on these definitions, we prove theoretically that there may exist some special kinds of computer viruses that have not been found in the real world yet. Furthermore, we prove that the set of computer viruses with the same kernel is $\prod_2$-complete. In general the set of computer viruses is $\sum_3$-complete.

Received 21 April 2003; revised 5 April 2004

1. INTRODUCTION

The first abstract theory of computer viruses is the viral set theory given by Cohen, based on the Turing machine [1, 2]. A viral set is defined by (M, V) where M is a Turing machine and V is a non-empty set of programs on Turing machine. Each $v \in V$ is called a computer virus and satisfies the following condition: if it is contained in the tape at time t, then there exist a time t' and a $v' \in V$ such that v' is contained in the tape at time t'. The most important of Cohen's theorems is about the undecidability of computer viruses [1].

In a different approach, Adleman developed an abstract theory of computer viruses based on recursive functions [3]. In his definition a virus is a total recursive function v which applies to all programs i (the Gödel numberings of programs), such that $v(i)$ has some characteristic behaviors of computer viruses like injury, infection and imitation. Furthermore, Adleman proved that the set of computer viruses is $\prod_2$-complete [3].

In the past 10 years, the number and variety of computer viruses have greatly increased, and many of them may not be conveniently and directly described by Cohen's or Adleman's theory. For example, we can list resident viruses, polymorphic viruses, stealthy viruses and so on. Hence some other models and theories of computer viruses have been established [4]. But these models do not comply with the common understanding of computer viruses.

The structure of this paper is as follows: the next section introduces preliminaries and recursive-function-based definitions for several kinds of computer viruses are presented in Section 3. In Section 4, we derive some important results about computer viruses based on the definitions. In the last section, we discuss the limitations of our work and give some comments and suggestions for further research.

2. PRELIMINARIES

We briefly present here basic notation used in this paper.

Let N be the set of all natural numbers and S be the set of all finite sequences of natural numbers. For every $s_1, s_2, \ldots, s_n \in S$, $\langle s_1, s_2, \ldots, s_n \rangle$ denotes a computable injective function from S^n to N with computable inverse. If f is a partial function $f : N \to N$, then for $s_1, s_2, \ldots, s_n \in S$, $f(s_1, s_2, \ldots, s_n)$ denotes $f(\langle s_1, s_2, \ldots, s_n \rangle)$. Similarly, for every $i_1, i_2, \ldots, i_n \in N$, $\langle i_1, i_2, \ldots, i_n \rangle$ denotes a computable injective function from N^n to N with computable inverse such that $\langle i, j \rangle \geq i$, and $f(i_1, i_2, \ldots, i_n)$ means $f(\langle i_1, i_2, \ldots, i_n \rangle)$ for partial function $f : N^n \to N$. Furthermore, we write $f(i_1, i_2, \ldots, i_n) \downarrow$ when $f(i_1, i_2, \ldots, i_n)$ is defined and $f(i_1, i_2, \ldots, i_n) \uparrow$ when $f(i_1, i_2, \ldots, i_n)$ is undefined.

For a sequence $p = (i_1, i_2, \ldots, i_k, \ldots, i_n) \in S$, we use $p[j_k/i_k]$ to denote the sequence which is the same as p except that i_k replaced by j_k, i.e. $p[j_k/i_k] = (i_1, i_2, \ldots, j_k, \ldots, i_n)$. If the element i_k in sequence p is operated by a computable function v, namely $p[v(i_k)/i_k]$, for conciseness of notation, it will always be written in this paper as $p[v(\underline{i_k})]$ where the underlined symbol denotes the element being operated. If there are more than one element in p that have been replaced by other values or operated by computable functions, we write them 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.

The symbol $\phi_P(d, p)$ denotes a function computed by a computer program P with the running environment (d, p) where d and p mean data (including clock, spaces of diskettes and so on) and programs (including operating systems), respectively. If the Gödel numbering of P is e, the function is commonly written as $\phi_e(d, p)$. Its domain and range are written as W_e and E_e, respectively.

S–m–n theorem, universal theorem and recursion theorem [5] are the main tools used in our development of the abstract theory of computer viruses.

3. DEFINITIONS OF SOME KINDS OF COMPUTER VIRUSES

In this section, we give definitions of some kinds of computer viruses, including not only some common kinds of computer viruses, e.g. non-resident viruses and resident viruses, but also some special kinds of computer viruses, e.g. polymorphic viruses with infinite forms which have not been found till now in the real world.

DEFINITION 3.1. (Non-resident virus) *A total recursive function v is called a non-resident virus if for all i,*

(a)
$$\phi_{v(i)}(d,p) = \begin{cases} D(d,p), & \text{if } T(d,p) \quad (i) \\ \phi_i(d, p[v(\underline{S(p)})]), & \text{if } I(d,p) \quad (ii) \\ \phi_i(d,p), & \text{otherwise} \quad (iii) \end{cases}$$

(b) $T(d,p)$ and $I(d,p)$ are two recursive predicates and no $\langle d,p\rangle$ satisfies them simultaneously; $D(d,p)$ and $S(p)$ are two recursive functions;

(c) the set $\{\langle d,p\rangle : \neg(T(d,p) \vee I(d,p))\}$ is an infinite set.

The two predicates, $T(d,p)$ and $I(d,p)$, are called injury condition (trigger) and infection condition, respectively. When condition $T(d,p)$ is satisfied, the virus executes the injury function $D(d,p)$, and when condition $I(d,p)$ is met, the virus chooses a program using the selection function $S(p)$, infects it first, and then executes the original program. These two conditions and two functions, called the kernel of a non-resident virus, determine a non-resident virus uniquely. In what follows, unless stated otherwise, the kernel of a computer virus always denotes the set of mathematical objects (functions and predicates) which theoretically determine a computer virus uniquely.

In clause (a) of the above definition, three branches (i), (ii) and (iii) describe three typical behaviors of computer viruses, injury, infection and imitation, respectively.

In the following definitions of other kinds of computer viruses, we no longer list (b) and (c) as in the definition above, and always regard them as satisfied by each kind of computer virus. It should be noted that the kernels of different kinds of computer viruses are different in general.

DEFINITION 3.2. (Resident virus) *The pair $(v, \mathbf{sys})$ of a total recursive function v and a system call* **sys** *(also a recursive function) is called a resident virus with respect to the system call* **sys** *if for all i,*

$$\phi_{v(i)}(d,p) = \begin{cases} D(d,p), & \text{if } T(d,p) \\ \phi_i(d, p[v(\underline{\mathbf{sys}})]), & \text{if } I(d,p) \\ \phi_i(d,p), & \text{otherwise} \end{cases}$$

and

$$\phi_{v(\mathbf{sys})}(d,p) = \begin{cases} D'(d,p), & \text{if } T'(d,p) \\ \phi_{\mathbf{sys}}(d, p[v(\underline{S(p)})]), & \text{if } I'(d,p) \\ \phi_{\mathbf{sys}}(d,p), & \text{otherwise} \end{cases}$$

Resident viruses always employ some system calls or endless execution processes (e.g. under Unix) to reside in the memory and modify some system calls (or some user processes) by which they infect other programs. For example, resident viruses in the DOS environment often modify system calls such as int 21h and int 13h to reach their objects.

DEFINITION 3.3. (Polymorphic virus with two forms) *The pair (v, v') of two different total recursive functions v and v' is called a polymorphic virus with two forms if for all i,*

$$\phi_{v(i)}(d,p) = \begin{cases} D(d,p), & \text{if } T(d,p) \\ \phi_i(d, p[v'(\underline{S(p)})]), & \text{if } I(d,p) \\ \phi_i(d,p), & \text{otherwise} \end{cases}$$

and

$$\phi_{v'(i)}(d,p) = \begin{cases} D(d,p), & \text{if } T(d,p) \\ \phi_i(d, p[v(\underline{S(p)})]), & \text{if } I(d,p) \\ \phi_i(d,p), & \text{otherwise} \end{cases}$$

Polymorphic viruses with n forms can be defined as sequences $(v_1, v_2, \ldots, v_n)$ of n different total recursive functions which satisfy similar conditions as the above definition. Polymorphic viruses have become transplanted over last 10 years and detecting them involves considerable difficulty. Commonly they have billions of forms and no two forms have the same consecutive three bytes in general. However, they are not the most difficult viruses to detect. In what follows, we define the polymorphic viruses with infinite forms, and prove their existence theoretically in the next section.

DEFINITION 3.4. (Polymorphic virus with infinite forms) *A total recursive function $v(m,i)$ is called a polymorphic virus with infinite forms if for all m, i,*

$$\phi_{v(m,i)}(d,p) = \begin{cases} D(d,p), & \text{if } T(d,p) \\ \phi_i(d, p[v(m+1, \underline{S(p)})]), & \text{if } I(d,p) \\ \phi_i(d,p), & \text{otherwise} \end{cases}$$

and for all $m \neq n$, $v(m,i) \neq v(n,i)$.

Polymorphic viruses with infinite forms have not actually been found so far, but their existence, similar to common computer viruses, is guaranteed by the same mathematical theorem (recursion theorem).

DEFINITION 3.5. (Stealthy virus) *The pair $(v, \mathbf{sys})$ of a total recursive function v and a system call* **sys** *is called a stealthy virus with respect to the system call* **sys** *if there is a recursive function h such that for all i,*

$$\phi_{v(i)}(d,p) = \begin{cases} D(d,p), & \text{if } T(d,p) \\ \phi_i(d, p[v(\underline{S(p)}), h(\underline{\mathbf{sys}})]), & \text{if } I(d,p) \\ \phi_i(d,p), & \text{otherwise} \end{cases}$$

and

$$\phi_{h(\mathbf{sys})}(x) = \begin{cases} \phi_{\mathbf{sys}}(y), & \text{if } x = v(y) \\ \phi_{\mathbf{sys}}(x), & \text{otherwise} \end{cases}$$

The crucial distinction between stealthy viruses and common viruses is that stealthy viruses not only infect programs as common viruses do, but also modify some system calls such that, when someone or the computer system uses these system calls to check programs, the infected program appears the same as if it were not infected.

DEFINITION 3.6. (Combinatorial virus) *The pair* (a, h) *of two total recursive functions* a *and* h *is called a combinatorial virus if for all* i,

$$\phi_{ah(i)}(d, p) = \begin{cases} D(d, p), & \text{if } T(d, p) \\ \phi_i(d, p[ah(\underline{S_1(p)}), h(\underline{S_2(p)})]), & \text{if } I(d, p) \\ \phi_i(d, p), & \text{otherwise} \end{cases}$$

and

$$\phi_{h(i)}(d, p) = \phi_i(d, p)$$

where the two total recursive functions a *and* h *are called the* activation *function and the* hiding *function, respectively.*

The combinatorial virus $(a, h_1, \ldots, h_n)$ with n hiding functions can be defined similarly to the above. The extraordinary feature of combinatorial viruses is that they imitate totally original programs and do not make any injury and infection unless their activation functions apply to them.

4. SOME THEOREMS ABOUT COMPUTER VIRUSES

In this section, we use some symbols listed as follows.

SYMBOLS 1.
$D_n = \{i : \phi_i \text{ is a non-resident virus}\}$
$D_r = \{\langle i, j\rangle : (\phi_i, \phi_j) \text{ is a resident virus}\}$
$D_p = \{\langle i_1, \ldots, i_n\rangle : (\phi_{i_1}, \ldots, \phi_{i_n}) \text{ is a polymorphic virus with } n \text{ forms}\}$
$D_i = \{i : \phi_i \text{ is a polymorphic virus with infinite forms}\}$
$D_s = \{\langle i, j\rangle : (\phi_i, \phi_j) \text{ is a stealthy virus}\}$
$D_c = \{\langle i, j\rangle : (\phi_i, \phi_j) \text{ is a combinatorial virus}\}$

If the superscript 'fixed' is attached to a symbol above, then it denotes the set of one kind of viruses which have a fixed kernel, e.g. D_r^{fixed} denotes the set of all resident viruses that have a fixed kernel. If its superscript is a name of a virus, e.g., $D^{\text{Jerusalem}}$, then it denotes the set of all viruses which have the same kernel as the Jerusalem virus.

THEOREM 4.1. *There exist polymorphic viruses with infinite forms.*

Proof. Let $b(m, i, k)$ be a 1-1 total recursive function such that

$$\phi_{b(m,i,k)}(d, p) = \begin{cases} \langle m, p\rangle, & \text{if } T(d, p) \\ \phi_i(d, p[\phi_k(m+1, \underline{S(p)})]), & \text{if } I(d, p) \\ \phi_i(d, p), & \text{otherwise} \end{cases}$$

Applying the s–m–n theorem to $b(m, i, k)$, there exists a total recursive function f such that

$$\phi_{f(k)}(m, i) = b(m, i, k).$$

By the recursion theorem, there is an n such that $\phi_{f(n)} = \phi_n$. Let $v(m, i) = b(m, i, n) = \phi_{f(n)}(m, i) = \phi_n(m, i)$, so

$$\phi_{v(m,i)}(d, p) = \phi_{b(m,i,n)}(d, p) = \begin{cases} \langle m, p\rangle, & \text{if } T(d, p) \\ \phi_i(d, p[\phi_n(m+1, \underline{S(p)})]), & \text{if } I(d, p) \\ \phi_i(d, p), & \text{otherwise} \end{cases} = \begin{cases} \langle m, p\rangle, & \text{if } T(d, p) \\ \phi_i(d, p[v(m+1, \underline{S(p)})]), & \text{if } I(d, p) \\ \phi_i(d, p), & \text{otherwise} \end{cases}$$

Notice that if $m \neq n$, then for all i, and all d, p such that $T(d, p)$ holds,
$\phi_{v(m,i)}(d, p) = \langle m, p\rangle \neq \langle n, p\rangle = \phi_{v(n,i)}(d, p)$
i.e. $v(m, i) \neq v(n, i)$. □

THEOREM 4.2. *There exist combinatorial viruses.*

Proof. Let h be a recursive function such that $\phi_{h(i)}(d, p) = \phi_i(d, p)$. Applying the s–m–n theorem, there exists a total recursive function $b(i, k)$ such that

$$\phi_{b(i,k)}(d, p) = \begin{cases} D(d, p), & \text{if } T(d, p) \\ \phi_i(d, p[\phi_k(h(\underline{S_1(p)})), h(\underline{S_2(p)})]), & \text{if } I(d, p) \\ \phi_i(d, p), & \text{otherwise} \end{cases}$$

By the recursion theorem, as in Theorem 4.1, there exists a total recursive function v such that

$$\phi_{v(i)}(d, p) = \begin{cases} D(d, p), & \text{if } T(d, p) \\ \phi_i(d, p[v(h(\underline{S_1(p)})), h(\underline{S_2(p)})]), & \text{if } I(d, p) \\ \phi_i(d, p), & \text{otherwise} \end{cases}$$

Let $a = v$, substituting i by $h(i)$ in the above equation, it follows that

$$\phi_{ah(i)}(d, p) = \begin{cases} D(d, p), & \text{if } T(d, p) \\ \phi_{h(i)}(d, p[a(h(\underline{S_1(p)})), h(\underline{S_2(p)})]), & \text{if } I(d, p) \\ \phi_{h(i)}(d, p), & \text{otherwise} \end{cases} = \begin{cases} D(d, p), & \text{if } T(d, p) \\ \phi_i(d, p[a(h(\underline{S_1(p)})), h(\underline{S_2(p)})]), & \text{if } I(d, p) \\ \phi_i(d, p), & \text{otherwise} \end{cases}$$

□

In fact, we proved that there exists a combinatorial virus (a, h) for any padding or compressing program h in Theorem 4.2.

The proofs of existence of other viruses are similar to the proof above, thus we omit them here. One thing necessary to know is that proofs of existence of resident viruses, polymorphic viruses and stealthy viruses will make use of the double(or multi-) recursion theorem.

THEOREM 4.3. *The set* D_n^{fixed} *is a* $\prod_2$*-complete set.*

Proof. Suppose recursive predicates $T(d, p)$ and $I(d, p)$ are the injury condition and the infection condition, and recursive functions $D(d, p)$ and $S(p)$ are the injury function and the selection function, respectively. Then, since

$$
\begin{aligned}
& e \in D_n^{\text{fixed}} \\
& \quad \Longleftrightarrow \phi_e \text{ is a total recursive function} \\
& \quad \bigwedge \forall i \forall d, p \\
& \quad \Big[\big[T(d, p) \to \phi_{\phi_e(i)}(d, p) = D(d, p)\big] \\
& \quad \vee \big[I(d, p) \to \phi_{\phi_e(i)}(d, p) = \phi_i(d, p[\phi_e(\underline{S(p)})])\big] \\
& \quad \vee \big[\neg(T(d, p) \vee I(d, p)) \to \phi_{\phi_e(i)}(d, p) = \phi_i(d, p)\big]\Big]
\end{aligned}
$$

and 'ϕ_e is a total recursive function' is a $\prod_2$- predicate, and '$\phi_x(z) = \phi_y(z)$' is a $\sum_1$-predicate, so '$e \in D_n^{\text{fixed}}$' is a $\prod_2$-predicate.

To prove D_n^{fixed} is a $\prod_2$-complete set, without loss of generality, we suppose that viruses do not make any injury and their infection condition is $I(d, p)$.

Let A be an $\prod_2$-set, then there exists a recursive predicate R such that $x \in A \Leftrightarrow \forall y \exists z R(x, y, z)$.

Consider the function

$$
f(i, k, x, \langle d, p\rangle) = \begin{cases} \phi_i(d, p[\phi_k(\underline{S(p)})]), & \text{if } I(d, p) \\ \phi_i(d, p), & \text{if } \neg I(d, p) \wedge \forall y < \langle d, p\rangle \exists z R(x, y, z) \\ \uparrow, & \text{otherwise} \end{cases}
$$

By Church's thesis, it is a recursive function: suppose $P(x, y, z)$ is the procedure computing the characteristic function of predicate $R(x, y, z)$. For a given $(i, k, x, \langle d, p\rangle)$, first test the recursive predicate $I(d, p)$; if it is true, then compute $\phi_i(d, p[\phi_k(\underline{S(p)})])$; if it is false, then compute $P(x, 0, 0), \ldots, P(x, \langle d, p\rangle, 0)$, $P(x, 0, 1), \ldots, P(x, \langle d, p\rangle, 1), \ldots$. If for each $y < \langle d, p\rangle$, there exists an n in this sequence such that $P(x, y, n) = 1$, then stop the computing procedure to compute $\phi_i(d, p)$; otherwise, the procedure cannot stop, so function $f(i, k, x, \langle d, p\rangle) \uparrow$.

Applying the *s–m–n* theorem to $f(i, k, x, \langle d, p\rangle)$, there exists a recursive function $b(i, k, x)$ such that $\phi_{b(i,k,x)}(d, p) = f(i, k, x, \langle d, p\rangle)$. By the recursion theorem with parameters, there exists a recursive function $n(x)$ such that $\phi_{n(x)}(i) = b(i, n(x), x)$, i.e.

$$
\begin{aligned}
& \phi_{\phi_{n(x)}(i)}(d, p) \\
& \quad = \phi_{b(i,n(x),x)}(d, p) \\
& \quad = \begin{cases} \phi_i(d, p[\phi_{n(x)}(\underline{S(p)})]), & \text{if } I(d, p) \\ \phi_i(d, p), & \text{if } \neg I(d, p) \wedge \forall y < \langle d, p\rangle \exists z R(x, y, z) \\ \uparrow, & \text{otherwise} \end{cases}
\end{aligned}
$$

Thus, if $x \in A$, then

$$
\begin{aligned}
x \in A & \Rightarrow \forall y \exists z R(x, y, z) \\
& \Rightarrow \phi_{\phi_{n(x)}(i)}(d, p) \\
& = \begin{cases} \phi_i(d, p[\phi_{n(x)}(\underline{S(p)})]), & \text{if } I(d, p) \\ \phi_i(d, p), & \text{otherwise} \end{cases} \\
& \Rightarrow n(x) \in D_n^{\text{fixed}}
\end{aligned}
$$

On the other hand, assume $x \notin A$, then $\exists y \forall z \neg R(x, y, z)$. Since, for each $e \in D_n^{\text{fixed}}$, there exist infinite numbers of $\langle d, p\rangle$ such that $\phi_{\phi_e(i)}(d, p) = \phi_i(d, p)$, it follows that there exists a large enough $\langle d', p'\rangle$ such that $\exists y < \langle d', p'\rangle \forall z \neg R(x, y, z)$, hence for all i, $\phi_{\phi_{n(x)}(i)}(d', p') \uparrow$. Assume ϕ_i is a total recursive function, then for all $e \in D_n^{\text{fixed}}$, $\phi_{\phi_e(i)}(d', p') = \phi_i(d', p')$ is defined at each $\langle d', p'\rangle$, hence $n(x) \notin D_n^{\text{fixed}}$.

According to the above discussion, it follows that

$$A \leq_m D_n^{\text{fixed}},$$

i.e. D_n^{fixed} is a $\prod_2$-complete set. □

In the following proof, we shall use the lemma below.

LEMMA 4.1. *For a given recursive set R, if its complement $\overline{R}$ is an infinite set, then there exists a recursive function g such that for all $\sum_3$-set A,*

$$
\begin{aligned}
& x \in A \Rightarrow [W_{g(x)} \textit{ is a recursive set }] \wedge [R \subseteq W_{g(x)}] \\
& x \notin A \Rightarrow [W_{g(x)} \textit{ is not a recursive set }] \wedge [R \subseteq W_{g(x)}]
\end{aligned}
$$

Proof. Let $W^0_{g(x)} = R$ in the proof of theorem 3.4 in chapter IV in [6]. □

THEOREM 4.4. *D_n is a $\sum_3$-complete set.*

Proof. By the definition of D_n, we can establish the following logical equivalence,

$$
\begin{aligned}
& e \in D_n \\
& \quad \Longleftrightarrow \phi_e \text{is a total recursive function} \\
& \quad \bigwedge \exists t \exists i \exists o \exists b \exists s \\
& \quad [[W_t, W_i, W_o \text{ are recursive sets}] \\
& \quad \wedge [W_t, W_i, W_o \text{ are pairwise disjoint}] \\
& \quad \wedge [W_t \cup W_i \cup W_o = N] \\
& \quad \wedge \forall x \forall d, p \\
& \quad [[\langle d, p\rangle \in W_t \to \phi_{\phi_e(x)}(d, p) = \phi_b(d, p)] \\
& \quad \vee [\langle d, p\rangle \in W_i \to \phi_{\phi_e(x)}(d, p) = \phi_x(d, p[\phi_e(\underline{\phi_s(p)})])] \\
& \quad \vee [\langle d, p\rangle \in W_o \to \phi_{\phi_e(x)}(d, p) = \phi_x(d, p)]]]
\end{aligned}
$$

where $\langle d, p\rangle \in W_t$, $\langle d, p\rangle \in W_i$ and $\langle d, p\rangle \in W_o$ correspond to $T(d, p)$, $I(d, p)$ and otherwise in the Definition 3.1 of non-resident viruses respectively, and recursive functions $\phi_b(d, p)$ and $\phi_s(d, p)$ denote $D(d, p)$ and $S(p)$, respectively.

Since 'ϕ_e is a total recursive function' and '$W_x = N$' are $\prod_2$-predicates, '$W_x = \emptyset$' is a $\prod_1$-predicate, '$\phi_x(z) =$

$\phi_y(z)$' is a $\sum_1$-predicate, and 'W_x is a recursive set' is a $\sum_3$-predicate, so $e \in D_n$ is a $\sum_3$-predicate.

Let R be an infinite recursive set whose complement set $\overline{R}$ is also infinite. Let $a \in N$ and $S(p)$ be the selection function. Consider the function

$$f(i,k,x,\langle d,p\rangle) = \begin{cases} \phi_i(d, p[\phi_k(\underline{S(p)})]), & \text{if } \langle d,p\rangle = a \\ \phi_i(d,p), & \text{if } \langle d,p\rangle \in W_{g(x)}\backslash\{a\} \\ \uparrow, & \text{otherwise} \end{cases}$$

where $g(x)$ is the function in Lemma 4.1. By Church's thesis, function $f(i,k,x,\langle d,p\rangle)$ is a recursive function. Applying the *s–m–n* theorem to it, there exists a recursive function $b(i,k,x)$ such that $\phi_{b(i,k,x)}(d,p) = f(i,k,x,\langle d,p\rangle)$. By the recursion theorem with parameters, there exists a recursive function $n(x)$ such that $\phi_{n(x)}(i) = b(i,n(x),x)$, thus

$$\begin{aligned}\phi_{\phi_{n(x)}(i)}(d,p) &= \phi_{b(i,n(x),x)}(d,p) \\ &= \begin{cases} \phi_i(d, p[\phi_{n(x)}(\underline{S(p)})]), & \text{if } \langle d,p\rangle = a \\ \phi_i(d,p), & \text{if } \langle d,p\rangle \in W_{g(x)}\backslash\{a\} \\ \uparrow, & \text{otherwise} \end{cases}\end{aligned}$$

Let A be a $\sum_3$-set, it follows from Lemma 4.1 and the above equation that

$$\begin{aligned} x \in A &\Rightarrow [W_{g(x)}\text{is a recursive set}] \wedge [R \subseteq W_{g(x)}] \\ &\Rightarrow W_{g(x)}\backslash\{a\} \text{ is an infinite recursive set} \\ &\Rightarrow n(x) \in D_n \end{aligned}$$

On the other hand,

$$\begin{aligned} x \notin A &\Rightarrow [W_{g(x)}\text{is not a recursive set}] \wedge [R \subseteq W_{g(x)}] \\ &\Rightarrow W_{g(x)}\backslash\{a\} \text{ is not an infinite recursive set} \\ &\Rightarrow n(x) \notin D_n \end{aligned}$$

Thus, it follows that D_n is a $\sum_3$-complete set. □

THEOREM 4.5. *The sets D_r^{fixed}, D_p^{fixed}, D_i^{fixed}, D_s^{fixed} and D_c^{fixed} are $\prod_2$-complete sets, and the sets D_r, D_p, D_i, D_s and D_c are $\sum_3$-complete sets.*

Proof. The proof is similar to Theorems 4.3 and 4.4. □

Theorems 4.3–4.5 mean that detecting viruses is quite intractable in the following sense: the degree of undecidability of the set of one kind of computer viruses which have same kernel, is 2 (e.g., $D_r^{\text{Jerusalem}}$); the degree of undecidability of the set of one kind of all computer viruses (e.g. D_r) is 3. Furthermore, in the proof of Theorem 4.4, if $x \notin A$, then the recursive function $n(x)$ is not a virus by our definition of viruses, because it has at least one branch in which condition is not a recursive predicate. Thus, the corollary below is obtained.

COROLLARY 4.1. *If the set of all computer viruses is a $\sum_3$-set, then it is a $\sum_3$-complete set.*

For a given virus v, let the set of all programs infected by v be $I_v = Rg(v) = \{v(x)|x \in N\}$ [3]. If I_v is a recursive set, then there exists a procedure deciding whether or not a particular program is infected by virus v. Whenever a program becomes infected by v, it can be detected by this procedure and removed from the computer, i.e. the virus v can be isolated from the computer environment. Set I_v is a recursively enumerable set certainly, but does not have to be a recursive set. The next theorem gives an instantial computer virus v such that I_v is not a recursive set. In other words, for this virus v, the strongest detecting procedure for it can just pick up every program infected by v, but cannot find all programs not infected by v.

Before giving the proof of the next theorem, we first state a lemma that will be used in the proof.

LEMMA 4.2. *There exists a total recursive function h such that $\phi_{h(i)}(d,p) = \phi_i(d,p)$ and E_h is a $\sum_1$-complete set.*

Proof. Let $j(i,x)$ be an increasing padding function, i.e. for all i and x, $\phi_{j(i,x)}(d,p) = \phi_i(d,p)$ (as in the proof of Theorem 4 of [3]). Let g be a total recursive function such that $\mathbf{K} = E_g$, consider the function

$$h(i) = \begin{cases} j(1, g(x)), & \text{if } i = j(1,x) \\ j(i, g(0)), & \text{otherwise} \end{cases}$$

It is clear that $\phi_{h(i)}(d,p) = \phi_i(d,p)$. Let $c(x) = j(1,x)$, since $j(i,x)$ is a 1-1 function, it follows that

$$x \in \mathbf{K} \Leftrightarrow c(x) \in E_h$$

i.e. $\mathbf{K} \leq_1 E_h$. Thus, E_h is a $\sum_1$-complete set. □

THEOREM 4.6. *There exists a computer virus v such that I_v is a $\sum_1$-complete set.*

Proof. Let h be the function satisfying conditions in Lemma 4.2. Applying the *s–m–n* theorem, let $b(i,k)$ be the 1-1 total recursive function such that

$$\phi_{b(i,k)}(d,p) = \begin{cases} D(d,p), & \text{if } T(d,p) \\ \phi_i(d, p[\phi_k(h(\underline{S(p)}))]), & \text{if } I(d,p) \\ \phi_i(d,p), & \text{otherwise} \end{cases}$$

By recursion theorem, there exists a function n such that

$$\phi_{n(i)}(d,p) = \begin{cases} D(d,p), & \text{if } T(d,p) \\ \phi_i(d, p[n(h(\underline{S(p)}))]), & \text{if } I(d,p) \\ \phi_i(d,p), & \text{otherwise} \end{cases}$$

Substituting i by $h(i)$ in the above equation, it follows that

$$\begin{aligned}\phi_{nh(i)}(d,p) &= \begin{cases} D(d,p), & \text{if } T(d,p) \\ \phi_{h(i)}(d, p[n(h(\underline{S(p)}))]), & \text{if } I(d,p) \\ \phi_{h(i)}(d,p), & \text{otherwise} \end{cases} \\ &= \begin{cases} D(d,p), & \text{if } T(d,p) \\ \phi_i(d, p[n(h(\underline{S(p)}))]), & \text{if } I(d,p) \\ \phi_i(d,p), & \text{otherwise} \end{cases}\end{aligned}$$

Let $v = nh$, then v is a non-resident virus. Since $n(x)$ is a 1-1 function, so $I_v = E_{nh}$ is not a recursive set. Otherwise, since

$$x \in E_h \Leftrightarrow n(x) \in E_{nh}$$

it follows that E_h is a recursive set and contradicts Lemma 4.2. □

The method proving Lemma 4.2 comes from Adleman's paper [3] and the method proving Theorem 4.6 can be used to prove that there exists any type of computer virus v such that I_v is a $\sum_1$-complete set.

5. DISCUSSION

Definition 3.1 of non-resident viruses, which we take as the typical definition of computer viruses, describes the characteristic behaviors of computer viruses. But clauses in the definition seem too restrictive, especially clauses (b) and (c), which deserve some further discussion here.

The clause (b) in Definition 3.1 requires that $T(d, p)$ and $I(d, p)$ are recursive predicates. It means that a computer virus must make injury and infection in some definite conditions (random activations of some computer viruses can be thought as definite functions of $\langle d, p\rangle$ mathematically). Computer viruses found up to now satisfy this restriction, but there may be computer viruses whose injury and infection conditions are semi-recursive predicates. This restrictive condition is necessary in the proof of Theorem 4.4, thus, if we can prove the theorem without it, then it can be removed from clause (b) in all definitions of computer viruses.

Clause (c) in Definition 3.1, which requires that an infected program imitate the original program at infinite points, is a quite strong condition (even though most computer viruses in the real world satisfy this condition) and excludes several rogue programs. The proof of Theorem 4.3 makes use of it. In fact, Theorem 4.3 can be proved in a weaker condition that the infected program has infinite points satisfying branches (ii) and (iii) in clause (a), namely, the set $\{\langle d, p\rangle : \neg T(d, p)\}$ is an infinite set. It is unknown whether Theorem 4.3 can be proved without clause (c) or not.

Another important fact that needs to be known in definitions of computer viruses is that there are two different ways of infection. One is infecting programs first and then executing the original program, i.e. $\phi_i(d, p[v(\underline{S(p)})])$, and another is executing the original program first and then infecting programs, i.e. $\phi_i(d, p)[v(\underline{S(p)})]$. We adopt the former in our definitions of computer viruses and there is no essential difference if we use the latter.

Corollary 4.1 looks strange at first glance because it was proved in Adleman's paper [3] that the set of all computer viruses is $\prod_2$-complete. This variation comes from the difference between our definitions and Adleman's definitions of computer viruses. It is known from the proof of Theorem 4.4 that the reason that D_n is $\sum_3$-set is that there exist recursive sets W_t, W_i and W_o which satisfy some conditions, but this requirement is not needed in Adleman's definitions of computer viruses.

We only define some familiar and interesting kinds of computer viruses in Section 3, however, other kinds of computer viruses can be defined easily in this way (sometimes, slight modification is needed). We give illustrative definitions of some other kinds of computer viruses as follows.

DEFINITION 5.1. (Non-resident overwriting virus) *A total recursive function v is called a non-resident overwriting virus if for all i,*

$$\phi_{v(i)}(d, p) = \begin{cases} D(d, p), & \text{if } T(d, p) \quad (i) \\ \langle d, p[v(\underline{S(p)})]\rangle, & \text{otherwise} \quad (ii) \end{cases}$$

A nonresident overwriting virus does not imitate the original program. When injury condition $T(d, p)$ is met, it causes damage; otherwise, it selects a program and overwrites it by itself. In this sense, non-resident overwriting viruses still satisfy the requirement in clause (b) in Definition 3.1 that no $\langle d, p\rangle$ satisfies $T(d, p)$ and $I(d, p)(= \neg T(d, p))$ simultaneously.

DEFINITION 5.2. (Interconvertible virus) *The pair (v, v') of two different total recursive functions v and v' is called an interconvertible virus if for all i,*

$$\phi_{v(i)}(d, p) = \begin{cases} D(d, p), & \text{if } T(d, p) \\ \phi_i(d, p[v'(\underline{S(p)})]), & \text{if } I(d, p) \\ \phi_i(d, p), & \text{otherwise} \end{cases}$$

and

$$\phi_{v'(i)}(d, p) = \begin{cases} D'(d, p), & \text{if } T'(d, p) \\ \phi_i(d, p[v(\underline{S'(p)})]), & \text{if } I'(d, p) \\ \phi_i(d, p), & \text{otherwise} \end{cases}$$

where $T(d, p)$(resp. $I(d, p)$, $D(d, p)$, $S(p)$) is different from $T'(d, p)$(resp. $I'(d, p)$, $D'(d, p)$, $S'(p)$).

An interconvertible virus (v, v') looks like it consists of two different computer viruses v and v'. But, there is a crucial distinction that when v infects a program, the result is infected by v'(not v), and vice versa. The interconvertible virus $(v_1, v_2, \ldots, v_n)$ can be defined similarly. The main dissimilarity between interconvertible viruses and polymorphic viruses is that each form of a polymorphic virus has the same kernel, but each component v_n of an interconvertible virus has its own different kernel.

DEFINITION 5.3. (Compositive virus) *The pair (v_1, v_2) of two different computer viruses v_1 and v_2 is called a compositive virus if and only if v_1v_2 is a computer virus too, namely for all i, v_1,v_2 and v_1v_2 satisfy the following equations respectively,*

$$\phi_{v_1(i)}(d, p) = \begin{cases} D_1(d, p), & \text{if } T_1(d, p) \\ \phi_i(d, p[v_1(\underline{S_1(p)})]), & \text{if } I_1(d, p) \\ \phi_i(d, p), & \text{otherwise} \end{cases}$$

and

$$\phi_{v_2(i)}(d, p) = \begin{cases} D_2(d, p), & \text{if } T_2(d, p) \\ \phi_i(d, p[v_2(\underline{S_2(p)})]), & \text{if } I_2(d, p) \\ \phi_i(d, p), & \text{otherwise} \end{cases}$$

and

$$\phi_{v_1v_2(i)}(d,p)=\begin{cases}D(d,p), & \text{if } T(d,p)\\ \phi_i(d,p[v_1v_2(\underline{S(p)})]), & \text{if } I(d,p)\\ \phi_i(d,p), & \text{otherwise}\end{cases}$$

where $T_1(d,p)$ *(resp.* $I_1(d,p)$, $D_1(d,p)$, $S_1(p)$*),* $T_2(d,p)$ *(resp.* $I_2(d,p)$, $D_2(d,p)$, $S_2(p)$*) and* $T(d,p)$ *(resp.* $I(d,p)$, $D(d,p)$, $S(p)$*) are different from each other, namely, viruses* v_1, v_2 *and* v_1v_2 *have different kernels.*

Compositive viruses resemble combinatorial viruses in the sense that they both create a new virus when their components meet in the computer system. The difference between them is that the components of a compositive virus are viruses themselves whilst the components of a combinatorial virus are not.

DEFINITION 5.4. (Multipartite virus) *A total recursive function* v *is called a multipartite virus if for all* i*,*

$$\phi_{v(i)}(d,p)=\begin{cases}D(d,p), & \text{if } T(d,p)\\ \phi_i(d,p[v(\underline{\mathbf{MBR}})]), & \text{if } I_1(d,p)\\ \phi_i(d,p[v(\underline{S(p)})]), & \text{if } I_2(d,p)\\ \phi_i(d,p), & \text{otherwise}\end{cases}$$

and

$$\phi_{v(\mathbf{MBR})}(d,p)=\begin{cases}D'(d,p), & \text{if } T'(d,p)\\ \phi_{\mathbf{MBR}}(d,p[v(\underline{\mathbf{sys}})]), & \text{otherwise}\end{cases}$$

and

$$\phi_{v(\mathbf{sys})}(d,p)=\begin{cases}D''(d,p), & \text{if } T''(d,p)\\ \phi_{\mathbf{sys}}(d,p[v(\underline{S''(p)})]), & \text{if } I''(d,p)\\ \phi_{\mathbf{sys}}(d,p), & \text{otherwise}\end{cases}$$

where **MBR** *and* **sys** *denote* master boot record *and a* system call*, respectively.*

Multipartite viruses use a combination of techniques including infecting documents, executables and boot sectors to infect computers. The above definition is that of a typical multipartite virus. Each of its infected programs infects MBR (under condition $I_1(d,p)$) or a selected program (under condition $I_2(d,p)$). The infected MBR employs a system call to reside in memory when booting and then infect other programs. Multipartite viruses are examples of computer viruses which have more than one spreading mode and can be described similarly.

We may define almost all kinds of computer viruses in this way, however, there are some special kinds of computer viruses that are still difficult to describe. For example, there may be a computer virus that does nothing but modify a .C file on computer. After the user compiles and links the .C source file, the infected program is workable. Extending our definitions of computer viruses so as to describe all kinds of computer viruses is one of our further research works.

ACKNOWLEDGEMENTS

Prof. Qingxin Zhu read the whole paper and helped us with the English for which the authors are grateful. The referees made useful comments that improved the presentation of the paper enormously.

REFERENCES

[1] Cohen, F. (1989) Computational aspects of computer viruses. *Comput. Security*, **8**(4), 325–344.

[2] Cohen, F. (1994) *A Short Course on Computer Viruses*. Wiley.

[3] Adleman, L. M. (1988) An abstract theory of computer viruses. In Goldwasser, S. (ed.), *Advances in Cryptology (CRYPTO'88)*, LNCS 403, pp. 354–374. Springer-Verlag, Berlin.

[4] Thimbleby, H., Anderson, S. and Cairns, P. (1999) A framework for modelling trojans and computer virus infection. *Comput. J.*, **41**, 444–458.

[5] Rogers, H., Jr (1967) *Theory of Recursive Functions and Effective Computability*. McGraw-Hill.

[6] Soare, R. I. (1987) *Recursively Enumerable Sets and Degrees*. Springer-Verlag.