Viruses in Turing's Garden

Jean-Yves Marion

Viruses in Turing's Garden

by Jean-Yves Marion

Cohen and his supervisor Adleman defined a virus as follows: "A virus is a program that is able to infect other programs by modifying them to include a possibly evolved copy to itself". This definition seems to be well accepted by the computer security community as a foundational definition. Thus, a virus is a self-replicating program, whose offspring may be a mutation of the original program. Viruses thrive in our computers, which are based on Turing's model of computation. We discuss the fundamental reasons for this.

One of Turing's main achievements is the construction of a Universal Turing Machine (UTM), which corresponds to a self-interpreter. That is a UTM executes a program by reading and interpreting a data. Thus, a program is a data, and this feature is one of the keys to build self-replicating programs. Programs called quines, which generate an exact copy of their source code, provide an amazing illustration. To devise a self-replicating program, we can follow Thompson's Turing award lecture or use Kleene's second recursion theorem. Indeed, from a programming view, a virus may be defined by a specification such as the following:

```
Viral_Spec (virus vs)
        Send confidential information
        For each pdf file f, append f to vs
```

A virus, satisfying the above specification, will first steal some information and then infect all pdf files in the system by appending a copy of itself at the end of each file. Thus, a solution of this specification is a program v such that v is a fixpoint of Viral_Spec, i.e. v=Viral_Spec(v). We see that a solution v behaves like a virus with respect to Cohen's definition. Anyone familiar with Smullyan's works will have no difficulty seeing that such a specification is self-referential. Indeed vs refers to the virus defined by the specification. There are many methods now known of achieving self-reference specification, but Kleene's demonstration is constructive and so gives an explicit solution, that is a virus. Moreover, his demonstration is uniform with respect to a self-reference specification, and therefore, it provides a virus compiler, which outputs a virus from a specification of it.

Viruses, that we have developed above, spread between programs without changing their program: There is no mutation. However, virus program mutation is an important feature used to avoid anti-virus detection. Again, Kleene's second recursion theorem gives us an effective construction of viruses whose descendants are all different. Indeed, a specification (and more generally a computable function) has an infinite number of fixpoints. Although the set of fixpoints is not computable, we may construct an infinite subset of fixpoints. Each fixpoint of this subset is a solution of the same self-reference specification, and each fixpoint corresponds to a virus. Then we can build a virus, which mutates when it duplicates, so no two copies are the same. Consequently, viruses cannot be detected.

In addition to self-replication and mutation, self-modification - the ability of a program to modify itself - is also an important feature. . A self-modifying program is able to alter its own code. For example, it may change its instructions on the fly, or generate codes from data that it can access, and thus the overall program may be seen as a sequence of different code layers. As a result, we can say that "a data is a program". From a theoretical point of view, a model of computation of self-reproducing systems must reflect the crucial fact that a data can be activated and executed, such as in random access stored program machines of Hartmanis.

In summary, Cohen's or Adleman's model of viruses bears a resemblance to self-reproducing machines. Universal Turing machines and to the second recursion theorem of Kleene form the core of the study of self-reproducing machines. There are other directions worth exploring: We may think that from one generation to another, viruses evolve and do not compute the same thing. In practice, this happens when a virus is updated either to apply a patch or to embed new functionalities. Another direction might be inspired by works on synthetic models of living organisms, beginning with Turing's paper on morphogenesis and von Neumann and Burks' construction of a self-replicating cellular automaton.

References/Links

M. A. Arbib. From universal turing machines to self-reproduction. In A half-century survey on The Universal Turing Machine, pages 177–189. Oxford University Press, Inc., 1988. URL http://dl.acm.org/citation.cfm?id=57249.57255.

F. Cohen. Computer viruses: Theory and experiments. Computers & Security, 6 (1):22–35, 1987. ISSN 0167-4048. doi: http://dx.doi.org/10.1016/0167-4048(87) 90122-2.

S. C. Kleene. On notation for ordinal numbers. The Journal of Symbolic Logic, 3 (4):150–155, 1938. URL http://dx.doi.org/10.2307%2F2267778.

Jean-Yves Marion. From Turing machines to computer viruses. Phil. Trans. R. Soc. A. July 28, 2012; doi:10.1098/rsta.2011.0332

M. Turing. On computable numbers, with an application to the Entscheidungsproblem. In Proceedings of the London Mathematical Society. Second Series Turing [1937], pages 230–265.

A. M. Turing. The chemical basis of morphogenesis. Philosophical transactions of the Royal Society of London Series B, Biological sciences, B 237(641):37–72, 1952. http://turing.ecs.soton.ac.uk/browse.php/B/22.

Please contact:
Jean-Yves Marion
Université de Lorraine, LORIA, France
E-mail: Jean-Yves.Marion@loria.fr