Reply to 'Comment on "A Framework for Modelling Trojans and Computer Virus Infection"' by E. Mäkinen

H. THIMBLEBY[1], S. O. ANDERSON[2] AND P. A. CAIRNS[1]

[1]*School of Computing Science, Middlesex University, Bounds Green Road, London N11 2NQ, UK*
[2]*Division of Informatics, King's Buildings, University of Edinburgh, Edinburgh EH9 3JZ, UK*
Email: harold@mdx.ac.uk, soa@dcs.ed.ac.uk, p.cairns@mdx.ac.uk

Computer viruses are a worrying real-world problem, and a challenge to theoretical modelling. In this issue of the *Computer Journal*, Erkki Mäkinen proposes universal Turing machines in a critique of an earlier paper, 'A framework for modelling Trojans and computer virus infection' (Thimbleby, H., Anderson, S. O. and Cairns, P. (1998) *Comp. J.*, 41, 444–458). This short paper is a reply by those authors.

Received 3 March 2001

F. B. Cohen developed computer viruses and an initial theory [1]. One criterion for judging the adequacy of a theoretical model of computer viruses is the extent to which it might help practitioners address some of the issues that arise in practice. Cohen's model aided in the determination of the decidability issues of viruses. However, early theoretical models failed to aid in the discovery of mechanisms that inhibit the spread of viruses. This motivated our own attempt [2] to define a new framework to explicitly address the wide variety of virus behaviours and to help in the exploration of antivirus mechanisms and architectures. In turn our paper has stimulated E. Mäkinen's paper [3], 'Comment on "A framework for modelling Trojans and computer virus infection"', which precedes this reply.

The explicit starting point of Mäkinen's paper is that Turing machines (TMs), and in particular universal Turing machines (UTMs), can 'serve as a basis of a model illustrating the properties of viruses and other malicious programs.' Mäkinen defines a virus as a description of a TM on the tape of a UTM, as sketched out in his Section 3. His approach is in contrast to ours, which avoided any concrete constructions on grounds of implementation bias. Mäkinen's model is appropriate for basic undecidability results, as he himself concludes, but his model does not address, for instance, viruses that replicate in other ways than as narrowly specified. He claims, without elaborating, that his universal model can be increased in concreteness, indeed he claims that 'there is no reason preventing us to sharpen the above model based on UTMs to any level of fine granularity.'

Let us take his model, based on [4]—which itself has been criticized for being too concrete [5]—seriously. A TM operates on its instantaneous description, which consists of the tape contents, tape head position, and the internal state. In the notation used in [3, 4], a TM is a tuple $\langle M, w, q\rangle$: in addition to the machine M itself that Mäkinen defines, there is a current state, $q \in Q$, and the tape contents $w = \alpha_1\alpha_2 \in (\Gamma \cup \{\epsilon\})^*$, with the tape head assumed to be positioned on the first cell of α_2. Mäkinen assumes a UTM (call it $\mathcal{U}$) with an encoding of a TM M on its tape. Since M operates over its tape contents w, the tape of $\mathcal{U}$ will be $M\alpha_1 q\alpha_2$.[1]

By definition, $\mathcal{U}$ does not change the representation of M on its tape, and the simulated TM M cannot change any part of the tape other than the substrings α_1 and α_2 since together they represent the entire tape accessible to M.

Mäkinen considers a virus to be a description on M's tape w 'of a TM whose simulation [. . . by $\mathcal{U}$] causes another description of a viral TM to appear to the tape of the UTM.' But $\mathcal{U}$, as defined, simulates no machine other than M, moreover there is no way to change the definition of M (i.e. its representation on $\mathcal{U}$'s tape) without assuming $\mathcal{U}$ is something other than a standard UTM. The representation M is not and cannot be 'infected' but must itself be running a well-defined program that, if it is to support the operations Mäkinen supposes, must amongst other things, search for TM representations in w and change them. Unless M is also universal, it will not be able to simulate any such machines, but if M is itself a correct universal machine, then the arguments apply *reductio ad absurdum*: defining

[1]This construction, based on [4], assumes the alphabets $(Q, \Gamma, \ldots)$ are disjoint and together form the alphabet of $\mathcal{U}$'s tape symbols. In fact $\langle M, w, q\rangle$ can be represented in many equivalent ways; e.g. $\mathcal{U}$ could be a multitape machine, with M, α_1, α_2 and q represented on separate tapes.

viruses that actually change the behaviour of a universal machine or any simulation run by a universal machine is never achieved. These problems are not insurmountable, but $\mathcal{U}$ somehow has to be an operating system for TMs; in fact it would have to be bigger in some sense if it is to model cross-computer virus infection. Ironically, real viruses typically infect quite ordinary programs, rather than directly affecting the behaviour of metaprograms such as M or $\mathcal{U}$. In short, while Mäkinen's definition of a virus looks superficially attractive, when it is more fully explored as a concrete description it raises problematic issues that demand intricate solutions—and one might suppose that sufficiently detailed schemes would be hard to relate to real world issues, where specific representations involving tapes are irrelevant.

Programs have names and Mäkinen suggests an implementation involving a bit string embedded in the tape encodings of programs: a malicious program 'contains the same bit string falsely naming the program.' He thus commits to the idea that the naming process is open to inspection by programs and (in the absence of any security mechanism) the details of the coding for names can be tampered with. Who is to say what 'falsely' means; and of two unequal names, which is which? If the semantics of naming are mere containment as a substring, there is anyway no relevant truth value involved. Thus Mäkinen is unwittingly invoking the environment and observer (who perhaps knows a 'true' binding of names)—or some comparable concepts. In contrast, our framework makes environments and observers explicit, thus allowing them to be analysed in as much detail as the conventional part of the model.

Having a notion of 'true naming' supposes there is a feasible way to confirm the names of programs, which is impossible unless the environment in which programs are named and executed is restricted (in ways that we showed can be defined [2]): this is the sort of insight one would want to start getting from a framework for reasoning about viruses. Moreover an observer that does not have unlimited computational resources and so cannot always discover differences between two supposedly identical environments (i.e. the correct one and one that contains the virus) is an important part of our framework. Without this it seems hard to model the idea that viral development may go on unnoticed for some time. Further, Mäkinen does not consider the long-term evolution of programs, where readily detectable destructive activity is delayed and intermediate forms of programs may not contain anything resembling a virus (e.g. because of encryption). His failure to address infection mechanisms in any detail means it is hard to explore mechanisms to prevent infection. This was a major goal of our framework.

For insight into computer viruses, we argued that speed and space are key concepts. First because real computation is so constrained, and secondly because by modelling these concepts we hoped to refine previous work. Furthermore, concrete models of replication are too narrow: not all real viruses oblige us by replicating exactly (as in Mäkinen's model). Although virus detection is undecidable in a suitably abstract model, would it be the case that virus detection is computationally feasible under more realistic assumptions (given that both the infection and the detection methods are similarly constrained, and that the virus can encrypt itself)?

Finally Mäkinen is content to say '. . . [such] Turing machines can be defined, but it is questionable whether the effects of such viruses are meaningful to measure in an abstract model.' Indeed, but unfortunately the effects of real viruses are meaningful to measure, as many of us will be only too well aware, and there is no reason for real viruses to conform to the concrete form that Mäkinen discusses. Mäkinen's approach gives only a very weak characterization of computer viruses and does not address infection mechanisms. As a consequence, it is impossible to account for a wide range of virus-like mechanisms. In particular those that respond more widely to the environment, evolving as they propagate, are multistage or multipart. Our own attempt was to provide a framework adequate to the study of a wide range of viral phenomena.

We avoided simulation on concrete TMs (universal or otherwise) precisely because reality is more interesting. Our precise notion of virus and infection has no real parallel in earlier formal models of viral infection nor in Mäkinen's account. We made the key points that (i) metaphorical models of computer viruses (e.g. as analogous to biological pathogens) were unreliable; (ii) a proper theoretical understanding of viruses should accommodate observation and computational complexity; and (iii) real viruses are not restricted to replicate themselves in trivial ways (replication does not have to be literal, nor one virus to one copy). The supporting arguments were, briefly, that a viral infection can subvert a system and in particular its abilities to recognize infection (hence the need for an observer); secondly—and this is a key point—neither virus nor observer has unrestricted computational resources with which to operate; finally, that real virus writers have no compunction to make theoreticians' lives easy—they can use encryption, recombining and any other devious techniques at their disposal.

Viruses are a complex phenomenon. Neither our paper nor Mäkinen's paper are the last word, and we hope both stimulate further work in this fascinating and important area.

REFERENCES

[1] Cohen, F. B. (1994) *A Short Course on Computer Viruses* (2nd edn). John Wiley, New York.

[2] Thimbleby, H., Anderson, S. O. and Cairns, P. (1998) A framework for modelling Trojans and computer virus infection, *Comp. J.*, **41**, 444–458.

[3] Mäkinen, E. (2001) Comment on 'A framework for modelling Trojans and computer virus infection', *Comp. J.*, **44**, 321–323.

[4] Hopcroft, J. E. and Ullman, J. D. (1979) *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA.

[5] Naur, P. (1993) Understanding Turing's universal machine—personal style in program description. *Comp. J.*, **36**, 351–372.