

REPRODUCER CLASSIFICATION USING THE THEORY OF AFFORDANCES: MODELS AND EXAMPLES

Matt WEBSTER, Grant MALCOLM

*Department of Computer Science,
University of Liverpool, Liverpool, UK
E-mail: {matt,grant}@csc.liv.ac.uk*

Abstract

We present a novel classification of reproducers based on Gibson's theory of affordances. Using affordances we are able to describe reproducer behaviour and determine how much the reproducer relies on external agency during its reproductive process. We give an informal overview of four main reproducer types, along with illustrative examples, and introduce the notions of active, passive and biactive reproducers for further subclassification. A formal definition of reproduction models is given, and we give three worked examples to show the application of this formalism to the classification of both artificial and natural reproducers. We introduce a modified version of the Type I conjecture from our earlier work, and conjecture that all reproducers can be categorised as "trivial" (Type IV) reproducers with the introduction of a "deity" into the model. We give a detailed overview of future work, with details on a potential application to computer virus detection.

Keywords: reproducers, classification, reproduction, affordances, artificial life, computer virology

1 Introduction

The classification of life, both natural and artificial, is relevant to several fields, including biology, artificial life and computer virology. In order to develop a classification of life forms, one must first determine what constitutes the class of living things. Defining the boundary between animate and inanimate often gives false positives and negatives. Reproduction, on the other

hand, is comparatively simple to define, and is an essential characteristic in most definitions of what it means to be alive (see, e.g., [23, 18]).

Reproducers are entities which engage in reproduction, where reproduction is the act of producing an offspring. Similar terms to reproduction exist in the literature, including “production”, “self-production”, “self-reproduction”, “replication” and “self-replication”. Some of these terms are used to distinguish between unaided and aided reproduction, however, we take “reproducer” to mean any entity which is reproduced, regardless of its level of involvement in the reproductive process.

In the literature there are clear and paradigmatic examples of reproducers: biological organisms and the genes that control them [6], von Neumann’s reproducing automaton [21], computer viruses [4] and other forms of reproducing malware [7], and so forth. However, there are other examples that stretch intuitive definitions of reproduction: photocopies [13], gliders in cellular automata [9], seeding crystals, fixed points of functions, or even a pen on a desk which, in being a static object, “reproduces” from one instant to the next thanks to the physical laws of the Universe.

One distinction between the paradigmatic and the trivial reproducers is that the members of the former group seem more independent and structured, whereas the latter group seem to reproduce almost accidentally, since the bulk of their reproductive processes is enabled by some external agency. One might try to construct a definition of reproduction that rules out such rogue examples; an alternative approach would be to classify different kinds of reproducers: i.e., to structure the space of reproducers in such a way that trivial and rogue examples of reproducers inhabit a clearly defined region that is separated from the region containing the paradigmatic examples. This is the approach taken in the present paper, where we give a classification of reproducers based on Gibson’s Theory of Affordances [10]. By classifying and structuring the space of reproducers, we hope to gain insight into the similarities and differences between various reproductive processes, and hence, if only indirectly, gain insight into what constitutes life.

The class of reproducers has been subdivided many times, according to various criteria (see, e.g., [16, 19]); a comprehensive overview of several of these classifications is given by Freitas & Merkle (ch. 5, [8]). Our approach seems to be novel in using a theory of affordances to develop an ecological approach to modelling reproduction. In Section 2, we set out our classification, which is based on a division between self-description on the one hand, and the machinery of reproduction on the other. Any specific example of a reproductive process will involve a number of agents (or *entities*), which will

include the reproducer itself. The classification is based on analysing which actions in the reproductive process are under the control of the reproducer itself, and which actions are made possible, i.e., afforded to the reproducer, by other entities.

Section 2 also gives types for each class of reproducer, e.g., von Neumann’s reproducing automaton, bacteriophage viruses, various kinds of computer viruses, and — in the trivial corner — photocopied documents. The approach in this section is very much a pragmatic realist one that discusses “real world” examples; Section 3 gives a more formalist approach to our classification, in which the focus is on classifying *models* of reproductive processes. We give a formal definition of such models, in which we can state precisely what is meant by an affordance. We also give worked examples of a computer virus, a bacteriophage virus and Langton’s loop. Although the presentation of the model of the computer virus is merely sketched, it is based on a very precise formal model presented as an equational theory of the semantics of assembly language [26, 24]. The two other formal reproduction models are constructed differently, based on an abstraction of the actions of the reproductive processes given by the empirical analysis of the behaviour of these reproducers.

This formal approach to modelling reproduction allows us to formulate relationships between models, which means that our classification is much more like a structured space of reproducers. This lets us develop formal statements of properties of the structure of this space, such as those given in Section 3.5.

This paper follows on from earlier work [27] on reproducer classification, in which we first laid out the means for reproducer classification using affordances. This paper gives more information and worked examples, and in Section 3.5 we introduce “self sets” as a means of modelling reproducers that evolve. In the same subsection we give new and modified conjectures on the structure of the reproducer space, and a potential application to computer virology. In the concluding section we give further directions for future research on this topic.

2 Classification Using Affordances

One of the earliest formal analyses of reproduction was given by von Neumann [21]. One of the aims of his work was to prove by construction the existence of a formalised automaton capable of reproduction. Von Neumann split his automaton into two separate parts: a self-description stored on a tape (analogous to a Turing machine tape), which was passed as input to a constructor capable of fabricating any configuration of cells within the cellular automaton

in which the reproducing automaton existed. This constructor was therefore a universal constructor; “universal” having an analagous meaning to “universal” Turing machines. Von Neumann’s reproducing automaton was based on a complex cellular automaton, but even the subsequent, and much more simple reproducers such as Langton’s [15] still retained an architecture based on a self-description and reproductive mechanism.

Our ontology and classification are based upon this division of an act of reproduction into self-description and reproductive mechanism phases. As well as von Neumann’s reproducing automaton and Langton’s loop, this abstract architecture is used by biological organisms, in the form of the genetic code (self-description) and the complex biological machinery which translates this code into offspring (reproductive mechanism). Biological viruses also use this architecture, although a significant part of their reproductive mechanism is given by an external entity: whichever organism the virus infects enables the reproduction of the virus, since viruses typically lack a sufficient reproductive machinery in order to complete their reproductive process. Other well-known reproducers such as Tierran organisms [17] generate a self-description by self-analysis. Their self-descriptions are not encoded as is the case with a genome, however the organisms still provide the description to themselves by their reproductive behaviour. The organisms copy their self-description one byte at a time to create an offspring, which constitutes the organism’s reproductive mechanism. Computer viruses and other forms of reproducing malware can generate a self-description in a similar way to the Tierran organisms (i.e., by self-analysis), or they can carry an encoded self-description within their own bytecode [26]. The reproductive mechanism is then the algorithm which creates a copy (offspring) of the virus based on the self-description. More “trivial” reproducers such as photocopies still employ a self-description and reproductive mechanism in their reproductive processes, although they are given by an external entity (the photocopier). Cellular automaton gliders such as those seen in Conway’s Game of Life [9] can also be seen to have a self-description and reproductive mechanism, although it is given to the reproducing gliders by an external agent: the self-description is contained in the state of the cellular automaton, and the reproductive mechanism is the state transition rule which creates the offspring (to be contained in the successive state). The reproductive architecture of self-description and reproductive mechanism can also be seen in other reproducers, but space limitations do not permit their inclusion.

The first step in classification involves identifying the parts of the reproductive process which correspond to the self-description and reproductive mechanism. Next we must determine which entities assist in the acts of self-descrip-

tion and reproductive mechanism. As we shall see in the forthcoming descriptions, definitions and worked examples, if the reproducer is not assisted in its reproductive process, i.e., if it is not afforded its self-description and reproductive mechanism by an external entity, then we classify it as “Type I”. A reproducer which is afforded either its reproductive mechanism or its self-description is either “Type II” or “Type III” respectively. A reproducer which is afforded both the self-description and reproductive mechanism by an external entity is “Type IV”.

Therefore, these types are the four corners of a reproducer classification space, and reproducers (such as biological organisms, artificial organisms, and computer viruses) occupy different points within this space depending on their reproductive reliance on themselves and the external agents in their environment.

2.1 Affordance Theory

Gibson’s theory of affordances describes the functional relationships between an animal and its environment [10]. For an animal, an affordance is an “opportunity for action.” For example, a piece of food affords an animal nourishment, a tree affords it the ability to climb to safety, and a cave affords shelter. We use affordances to form a classification of reproducers based on the functionality that they afford to themselves and the functionality that their environment affords them, with respect to reproduction. Gibson also describes a niche in an environment, into which an organism fits, as a “set of affordances.” Therefore, in this paper we categorise different reproducers according to the sets of affordances corresponding to their reproductive niches.

Here we use “affordance” metaphorically; we apply it to reproducers rather than animals, replacing “animal” with “reproducer” in Gibson’s original definition. If we allow this slight expansion of the definition of the word, this affordance-based ontology can afford us an intuitive and novel way of approaching the classification of reproducers.

For historical reasons “affordance” has come to mean “perceived affordance” in the domain of human–computer interaction [20]. It is important to note that apart from the slight expansion of the definition that we mention above, we take “affordance” to mean “opportunity for action”, as in the original definition given by Gibson.

2.2 Type I Reproducers

John von Neumann's reproducing automaton was the first mathematical model of a reproductive process. Von Neumann identified that reproduction is possible when the reproducer has a self-contained self-description, and a reproductive machinery that interprets the self-description as a set of instructions for producing an offspring [21]. From an abstract point of view, this is the same process that biological organisms undergo during reproduction, where the genome (qua self-description) is interpreted by the reproductive machinery within the organism to produce an offspring. We will call this mode of reproduction Type I. The von Neumann reproducer is completely self-reliant with respect to reproduction, as its reproductive mechanism and self-description are completely self-contained. The self-description consists of a tape that is attached to the automaton. The reproductive mechanism of the automaton consists of the method by which the tape is read and translated into a sequence of instructions, which are fed to the constructing arm in order to construct the offspring.

2.2.1 Example: Langton's Loop

Langton's reproducing loops consist of an outer sheath which encapsulates a sequence of cells with various states (a data signal), which effectively correspond to instructions to the construction arm for building an offspring loop [15]. The self-description here is the data signal within the sheath, and the reproduction mechanism is the means by which the instruction in the self-description are "interpreted" by the constructing arm. Both the self-description and reproductive mechanism are self-contained, and therefore this particular case of reproduction belongs in Type I.

2.3 Type II Reproducers

Biology, as well as artificial life and computer virology, presents us with examples of reproduction that do not fit into Type I. For example, biological viruses such as the T4 bacteriophage (Fig. 1) have a self-contained self-description, but lack a completely self-contained reproductive machinery to interpret the self-description and produce an offspring. Instead, these viruses inject their viral self-description into a host cell (e.g., a bacterium, in the case of T4) where it essentially hijacks the reproductive mechanism of the cell so that the cell no longer creates copies of itself, but rather copies of the virus.

A Type II reproducer differs from Type I in that it lacks a completely self-afforded reproductive mechanism. There is a self-afforded self-description

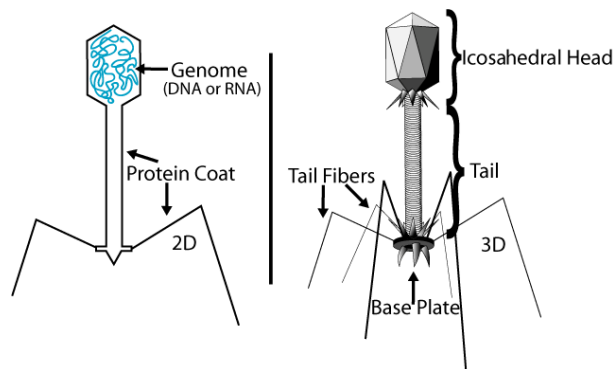


Figure 1. The T4 bacteriophage virus [14]. The legs of the T4 attach themselves to the cell wall of a bacterium. Then, using the base plate and tail as an injection mechanism, the genetic material in the head is transferred to the interior of the host cell, where it hijacks the reproductive mechanism of the cell in order to reproduce.

mechanism, but the reproductive mechanism is afforded (at least in part) by an external agent to the Type II reproducer.

2.3.1 Example: Automaton With Self-Description

An example of a Type II reproducer would be an automaton similar to the von Neumann reproducing automaton. This automaton does not need to be a universal constructor, or have any construction abilities at all. It must, however, contain a self-description such that when it is interpreted by an external agent with constructing abilities (i.e., an external reproductive mechanism) the resulting construction is an offspring of the reproducer. The Type II reproducer then reproduces by the agency of an external constructor that takes the self-description within the reproducer and creates an offspring based on it.

2.3.2 Example: T4 Bacteriophage

The T4 bacteriophage affords itself a sufficient self-description in the form of its genome, encoded as DNA or RNA (see Fig. 1). The first stage in the reproductive process (the injection of the genetic material into the host bacterium) is performed by the T4 itself, and therefore part of the reproductive mechanism is self-contained. However, since all of the subsequent stages of the reproduction are afforded by the bacterium, we can say the mode of reproduction for the T4 bacteriophage is Type II.

2.4 Type III Reproducers

So far we have described two reproductive types corresponding to the case where a self-description and a reproductive mechanism are afforded completely by a reproducer to itself (Type I), and the case where a self-description is afforded by the reproducer to itself but the reproductive mechanism is (at least partially) afforded by some other agent (Type II). Let us consider a reproducer that does not afford itself a self-description completely, but does afford itself a complete reproductive mechanism. Therefore, in order to complete its reproductive cycle it must obtain a complete self-description. We call this Type III reproduction.

Type III reproducers differ from Type I reproducers in that they lack a completely self-afforded self-description, but they do have a complete and sufficient self-afforded reproductive mechanism. In terms of the von Neumann reproducing automaton, we can visualise a Type III reproducer as being a von Neumann reproducing automaton without the input tape providing a self-description. The reproductive cycle can then only be completed by the action of an external agent that provides a self-description suitable for interpretation by the reproductive machinery of the reproducing automaton.

2.4.1 Example: Compiler

A compiler (in reproducing automata terms) is a constructor of programs, which takes as its input a sequence of symbols in some programming language (i.e., source code). A compiler can be given its own source code (self-description) as input, and as a result it will create a copy of itself based on these instructions. Therefore, a compiler has a completely self-afforded reproductive mechanism, but the self-description is afforded by external agency. Therefore a compiler is a Type III reproducer.

2.4.2 Example: Damaged Bacterium

It is difficult to think of a biological example approximating Type III reproduction. However, we might imagine a bacterium that has been damaged so that it no longer contains any genetic material. At this point there will be no self-description but there will be a completely self-afforded reproductive mechanism, and reproduction will only be possible through the action of an external agent that affords a sufficient self-description.

2.5 Type IV Reproducers

Let us now consider a reproducer that does not afford itself completely either a self-description or a reproductive mechanism. If reproduction is to take place it follows that some agent external to the reproducer must provide the necessary self-description and reproductive mechanism. We call this Type IV reproduction.

2.5.1 Example: Game of Life Gliders

Cellular automaton (CA) gliders in Conway's Game of Life [9] are able to reproduce trivially thanks to the transition rule of the CA. Here, the state of the CA at a particular instant stores the glider's description, and therefore acts as a self-description for the glider. The reproductive mechanism is provided by the transition rule which maps one state of the CA to the next, and causes the glider to reproduce to a new point on the CA grid. Therefore a glider is a Type IV reproducer, as it does not afford itself completely either a reproductive mechanism or a self-description.

2.5.2 Example: The Photocopy

The information stored in writing on a piece of paper is able to reproduce trivially in an environment with a photocopier. A piece of paper is fed into the photocopier, at which point it is scanned and a representation (digital or otherwise) which captures the appearance of the piece of paper is created. This representation is a self-description for the piece of paper. This representation is then fed to the printer within the photocopier, which creates a facsimile of the original piece of paper in the form of a photocopy. This printing process is the reproductive mechanism for the writing on the paper. The photocopy therefore does not afford itself completely either a self-description or reproductive mechanism, making it a Type IV reproducer.

2.6 Active, Passive and Biactive Reproducers

Reproducers outside Type I necessarily lack some essential part(s) of their reproduction mechanism: either the self-description, the reproductive mechanism, or both. These lacking parts must be provided by an external entity. However, some reproducers outside Type I may afford themselves an action which assists in the acquisition of a self-description and/or a reproductive mechanism in order to complete their reproductive process. For example, the

$rm \in Aff(r, r)$	Type III	Type I
$rm \in Aff(\hat{r}, r)$	Type IV	Type II
	$sd \in Aff(\hat{r}, r)$	$sd \in Aff(r, r)$

Figure 2. The four reproducer types, where r is a reproducer, \hat{r} is some entity other than r , sd and rm are the abstract actions corresponding to the self-description and reproductive mechanism respectively, and $Aff(r, r)$ is the set of actions that r affords itself. Note that affordance sets are not disjoint, so that (for example) if $sd \in Aff(\hat{r}, r)$ then $sd \in Aff(r, r)$ also, if needed.

T4 bacteriophage given in Section 2.3.2 has an injection mechanism for inserting its genetic code into a host cell. Reproducers like this are called *active*. Some reproducers, like the photocopy, do not afford themselves any such action obtaining a self-description or reproductive mechanism, and are therefore called *passive*. If a reproducer lacks both a self-description and a reproductive mechanism, but affords itself action(s) which enable its acquisition of both, then it is called *biactive*. Therefore, Type II and III reproducers can either be active or passive. Type IV reproducers can be either passive, biactive, or active with respect to the either the self-description or the reproductive mechanism.

3 A Formal Approach to Modelling Affordances

The preceding discussion has focused on the distinguishing characteristics of the different types and sub-types of reproduction. These characteristics were described in terms of the actions afforded by entities to other entities. The identification of the entities and actions involved in a process, and the allocation of affordances among these entities, depends upon the level of abstraction at which the process is viewed: in other words, what we are characterising is not so much processes themselves, as models of processes. A long-term goal of our research is not just to classify models of reproductive processes, but to be able to compare and relate different models. This might allow us, for example, to identify strategies for reproduction, such as the various ways of “hijacking” the reproductive machinery of other reproducers: bacteriophage viruses and computer viruses rely on another, general-purpose reproductive mechanism — proteins that will copy any DNA or RNA sequence; universal constructors or compilers that will work on any input sequence of instructions, and so forth. A deep understanding of such strategies would be useful, for example, in

computer virology, in designing security protocols for mobile processes. We believe that such advances are most likely to be achieved by providing formal notions of model and relationships between models, along the lines of the relationships between algebraic theories of formal ontologies (see, e.g., [2]).

In this section, we give a more formal approach to modelling reproduction and affordances. We begin by defining what we mean by a model of a reproductive process, and what an affordance is in such models. We then give worked examples of a copier computer virus (Section 3.2), a bacteriophage virus (Section 3.4) and Langton's loop (Section 3.3). Finally, Section 3.5 gives a discussion of how the space of reproduction models is structured by the introduction of two conjectures, and we introduce "self sets" as a means of modelling evolution within our ontology.

3.1 Models and Affordances

We assume that any model of a reproductive process identifies the states of affairs within which the process plays itself out. In more formal processes, such as the von Neumann reproducing automaton or computer viruses, these states of affairs may be very clearly and precisely defined: e.g., the states of the grid of cellular automata, or the states of a computer that contains a virus, including the files stored on disk, the contents of working memory, and so forth. The example we present in Section 3.2 below uses a very formal description of such computer states based on previous work in modelling computer viruses through an algebraic theory that captures the semantics of a computer assembly language [24, 26]. In more "real life" examples, such as the bacteriophage virus (3.4), these states may be more abstractly presented: e.g., some aqueous solution containing cells and viruses, with perhaps some virus attached to some cell membrane, or some cell having been injected with viral DNA, and so forth.

Two key elements of the states of a model are the entities that partake in the various states, and the actions that allow one state to evolve into another state. For example, we might consider some aqueous solution containing bacteria and bacteriophages, and we might identify a particular state in which a particular bacterium and a particular bacteriophage are present, and close to each other. If the bacteriophage attaches to the bacterium, this action takes us to a new state in which both the bacterium and bacteriophage are present, but now rather than merely being in the bacterium's neighbourhood, the bacteriophage is latched on to the cell's outer membrane. In general, we assume that a model identifies the key entities or agents that take part in the process being modelled, and has some way of identifying whether a particular entity

occurs in a particular state of affairs (e.g., once an infected bacterium's cell membrane has ruptured, that bacterium will presumably no longer be present in any further states). We also assume that a model identifies those actions that are relevant to the process being modelled, and describes which actions may occur to allow one state of affairs to be succeeded by another. In computer science, we call such a structure describing states, actions, and a relation of succession, a "labelled transition system"; as far as modelling is concerned, a key property is that these need not be deterministic: there might be several states that succeed another state as the result of some particular action.

This basic framework allows us to talk about reproductive processes: we can say that reproduction means that there is some entity r (the reproducer), some state s (the initial state of the reproductive process) with r present in state s (denoted " $r \varepsilon s$ " — see Definition 1 in this section) and some sequence $w = a_1, \dots, a_n$ of actions, such that w leads, through a succession of intermediate states, to a state s' with $r \varepsilon s'$. This, of course, allows for trivial reproductive processes, in which the entity r simply persists through the successive states from s to s' , but also allows for more interesting cases where r is, e.g., a von Neumann reproducing automaton, and the succession of states represents all the intermediate states involved in the construction of its copy. We assume that the relation $r \varepsilon s$ can be made abstract enough to accommodate an appropriate laxity in the notion of entity: i.e., we should gloss $r \varepsilon s$ as stating that the entity r , or a copy of r , or even a possible progeny of r , is present in the state s . In computer virology, such an abstraction was explicit in the pioneering work of Cohen [4], where a virus was identified with the *set* of forms that the virus could take. This approach is useful for so-called metamorphic viruses that, in an attempt to avoid detection, may mutate their source code. Cohen's sets of possible forms are referred to as "viral sets"; we might say that our approach identifies entities modulo "self sets." A more in-depth discussion of self sets is given in Section 3.5.3.

In the previous section, we saw that it was possible to characterise various types of reproductive systems, including trivial systems, by means of affordances. In the more formal approach of this section, we can say that affordances are ways of carving up possible actions among the entities of the system. Actions may be afforded by one entity to another. We write $Aff(e, e')$ for the actions that entity e affords to entity e' . The idea is that these are actions that are available to e' only in states where e is present. Thus, we require that a model carves up these actions in a coherent way: formally, for any state s where e' is present, the action a is possible (i.e., a leads to at least one state that succeeds s) only if e is also present in s .

These assumptions on models are captured in the following

Definition 1 A reproduction model *consists of*:

- a labelled transition system (S, A, \mapsto) , where S is a set of states, A is a set of actions (labels), and \mapsto is a ternary relation for labelled transitions between states, s.t. if $s \xrightarrow{a} s'$, the action a occurring in the state s leads to the new state s' ;
- a set Ent of entities and a relation $_ \varepsilon _$ between entities and states, where for $e \in Ent$ and $s \in S$, $e \varepsilon s$ indicates that e is present in the state s ;
- a function Aff that assigns to two entities e and e' , a set $Aff(e, e')$ of possible actions, in such a way that if $a \in Aff(e, e')$, then for all states s with $e' \varepsilon s$, a is possible in s (i.e., $s \xrightarrow{a} s'$ for some state s') only if $e \varepsilon s$. Notionally, $Aff(e, e')$ is the set of affordances that e gives to e' ;
- two sets of actions A_{sd} and A_{rm} which correspond to the actions involved in the reproducer's self-description and reproductive mechanism respectively, such that $A_{sd}, A_{rm} \subseteq A$.

This basic framework allows us to characterise reproductive systems, and classify them according to the types given in the previous section. For example, we could first specify that there is some reproducer r whose reproductive process can be described using a labelled transition system $s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \dots \xrightarrow{a_n} s_n$. In order to show that r is a Type I reproducer, we must specify that all actions are only afforded by the reproducer to itself, i.e., for each action a_i ($1 \leq i \leq n$), we have $a_i \in Aff(r, r)$. Therefore, every action in the reproductive process is only afforded by r to itself, and this includes the actions that correspond to the abstract actions sd and rm (i.e., the actions in A_{sd} and A_{rm} respectively). Therefore, $sd, rm \in Aff(r, r)$ and therefore r is Type I.

Type II classification would be as follows. We need to establish which actions correspond to the self-description and reproductive mechanism, and construct sets A_{sd} and A_{rm} respectively. Then, if all actions corresponding to the self-description are afforded only by the reproducer to itself, i.e., for all $a \in A_{sd}$, $a \in Aff(r, r)$, then we know that the reproducer is either Type I or II (see Fig. 2). If we also know that there is at least one action that is afforded by an external entity, e which is different from r , then we know that the reproducer r is Type II. Formally we say that there must exist some $a \in A_{rm}$ and $e \in Ent$ such that $a \in Aff(e, r)$, and therefore $\{r, e\} \subseteq Ent$. Type III classification is similar to Type II classification, except that sd and rm are swapped.

Type IV classification can be achieved by showing that there are actions in the reproductive process that are afforded by some external entity e to the reproducer r . Formally we say that there must exist some $a \in A_{rm}$, $a' \in A_{sd}$ and $e, e' \in Ent$ such that $a \in Aff(e, r)$ and $a' \in Aff(e', r)$, where e and e' are different from r .

3.2 Worked Example: Copier Computer Virus

In order to illustrate a concrete example of reproducer classification, we present the copier computer virus (Fig. 3). The virus shown here is written in SPL, an ad hoc programming language designed specifically for the purpose of modelling computer viruses [24]. The syntax and semantics of SPL are defined formally using OBJ, a formal notation for algebraic specification [11].

The copier virus exists in an environment consisting of a file system and an operating system that can manipulate the file system. Programs written in SPL can manipulate the file system using operating system function calls.

The behaviour of the computer virus is as follows. In line 1 the variable `fh1` is assigned the returning value of the function `getFileHandle`, which returns an arbitrary file handle from the file list. In line 2 the function `getSelfName` is called in order to return the file handle of the file currently running (i.e., the file handle of “self”) and it is assigned to variable `myName`. In line 3 the variable `nfh` is set to the value of a new file handle, that is, a file handle that is not currently being used by any other file. This is so that a temporary file can be created and written to during the infection process. In lines 4–5 two variables that are needed for the forthcoming `do {_} while (_)` loop are initialised.

In line 6 we encounter the loop. In the first iteration of the loop, the 0th statement (i.e., the first line) of the file named `myName` (which is the file currently running and therefore the file containing the virus) is read in by the function `getLine(_, _)` and assigned to the variable `line`. Next, this statement is written to the temporary file called `nfh` using the `writeToFile(_, _)` function. In line 9 the variable `counter` is incremented, so that on the next iteration of the loop the following line will be read in and written to the file `nfh`, and so on.

The net effect of this loop is that a copy of the virus is placed in a temporary file (`nfh`). The loop stops when the statement that has just been copied (`line`) is equal to the value of variable `lastLine`, which is set to `label end`. `label end` is the last executable line of the virus program (line 12) and separates the virus from the rest of the infected executable. Clearly, in this case the virus exists in a file alone, but the purpose of the guard is to make sure

```

1      fh1 := getFileHandle ;
2      myName := getSelfName ;
3      nfh := newFileHandle ;
4      counter := 0 ;
5      lastLine := label end ;
6      do {
7          line := getLine(myName,counter) ;
8          writeToFile(line,nfh) ;
9          counter := s(counter)
        } while ( not(line == lastLine) ) ;
10     prepend(nfh,fh1) ;
11     deleteFile(nfh) ;
12     label end

```

Figure 3. Copier computer virus programmed in SPL.

that in future generations only the virus is copied and not the rest of the host executable.

Next comes a call to the function `prepend(nfh, fh1)`, which causes the statements corresponding to `nfh` to be added to the start of the file `fh1`, in the order they appeared in `nfh`. This is the most crucial stage of viral infection, where the virus attaches itself to the host. In line 11 the temporary file `nfh` is deleted from the file system.

The overall effect of running the above virus program is that the virus searches for another executable file in the file system, which it infects by prepending its own code to that of the executable.

3.2.1 Classifying the Copier Virus

We shall show how the copier virus can be classified using two different models (M and N) for the reproductive system. In model M the entities in the environment are the copier virus (cv) and the file store (fs), which contains a list of executable files together with their file handles. The functions provided by the operating system are taken for granted as the “laws of the universe” in which the copier computer virus resides, and are not considered as actions explicitly afforded by the OS in the same way that the laws of thermodynamics are not usually considered to be actions afforded by the Universe on biological reproducers.

In the second model N the entities are the copier virus (cv) and the operating system (os), which encapsulates the file store from Model M , as well as a number of operating system (OS) functions which are used to modify the file

store. The model differs from the first model in that the OS is no longer taken for granted as a part of the physical laws of the universe, but rather is viewed as an external agent which affords the computer virus help in the form of actions performed by its function calls.

Some statements in the program Ψ are afforded by the copier virus to itself. However, certain other statements could not execute without external agency, and these statements are therefore afforded by either the file store (model M) or the operating system (model N).

Let $\Psi = \psi_1, \psi_2; \dots; \psi_n$ represent the statement list corresponding to the copier virus algorithm written in SPL. Let the set of states of the environment S be the set of states of the computer system, which includes a file store. (Therefore, if the file store is updated, the state of computer system changes.) Then, the ternary relation of state succession \mapsto is defined as follows: $\forall \psi_x \in \Psi : s \xrightarrow{\psi_x} s'$. (The precise semantics of these statements is defined formally using OBJ [24].) Since Ψ is a list of statements, and each one modifies the state of the machine executing the instructions, it is important to note that $\forall \psi_x \in \Psi : s \xrightarrow{\psi_x} s'$ iff $[[\psi_x]](s) = s'$ where $[[\psi_x]](s)$ is the effect of ψ_x on store s , which is to say that a state transition from s to s' under action ψ_x is only possible if the effect of ψ_x on state s is s' . The set of actions A consists only of the statements used by the copier virus algorithm, and therefore $A = \{\psi_1, \psi_2, \dots, \psi_n\}$. We can now assign different actions to different affordance sets for models M and N .

3.2.2 Classifying the Copier Virus Using Model M

Model M has two entities: the copier virus and the file store. Therefore, $Ent_M = \{cv, fs\}$. Numbering the twelve statements from ψ_1 to ψ_{12} we can calculate the dependence on the operating system using set analysis. So, $A_{sd} = \{\psi_2\}$ since this is the statement used to obtain a file handle for acquisition of the self-description (i.e., SPL code) and $A_{rm} = \{\psi_8, \psi_{10}, \psi_{11}\}$, since these are the statements which take the self-description and create an offspring within one of the executable files within the file store. We define the statements aided by the file store as those which access the file store, i.e., $Aff(fs, cv) = \{\psi_1, \psi_2, \psi_3\}$. Now we can calculate $A_{rm} \cap Aff(fs, cv)$ and $A_{sd} \cap Aff(fs, cv)$ in order to specify the reliance on external agency (i.e., in this case, the file store) by the copier virus. So, $A_{rm} \cap Aff(fs, cv) = \emptyset$ and $A_{sd} \cap Aff(fs, cv) = \{\psi_2\}$. Therefore the copier virus is a Type III reproducer within Model M , since the defining characteristics of Type III are that the reproductive mechanism is not afforded by any external entity and at least some part of the self-description is afforded by an external entity.

3.2.3 Classifying the Copier Virus Using Model N

Model N has two entities: the copier virus and the operating system. Therefore $Ent_N = \{cv, os\}$. Numbering the statements from ψ_1 to ψ_{12} as before we can calculate the dependence on the operating system using set analysis. So, $A_{sd} = \{\psi_2\}$ and $A_{rm} = \{\psi_8, \psi_{10}, \psi_{11}\}$, for the same reasons as for Model M . We define the statements (actions) afforded by the operating system as those which use operating system functions, i.e., $Aff(os, cv) = \{\psi_1, \psi_2, \psi_3, \psi_7, \psi_8, \psi_{10}, \psi_{11}\}$. Now we can calculate $A_{rm} \cap Aff(os, cv)$ and $A_{sd} \cap Aff(os, cv)$ in order to specify the reliance on external agency (i.e., in this case, the operating system) by the copier virus. So, $A_{rm} \cap Aff(os, cv) = \{\psi_8, \psi_{10}, \psi_{11}\}$ and $A_{sd} \cap Aff(os, cv) = \{\psi_2\}$. Therefore the copier virus is a Type IV reproducer within Model N , since the defining characteristics of Type IV are that the reproductive mechanism and self-description are afforded (at least partially) by external agency.

3.2.4 Comparing Models M and N

We found that for Models M and N we can categorise the copier computer virus within Types III and IV respectively.

Therefore, by changing our model for the reproductive system, we can see that the actions afforded by the external agents vary, which in turn can modify the classification we give to reproducers. In the case of the copier computer virus, using the operating system as an external agent instead of the file store changed the categorisation from Type III (non-trivial) to IV (trivial). It seems inconsistent to categorise a reproducer like the copier computer virus in the same class as a photocopy or a glider, but this apparent paradox is a result of the choice of model, rather than a problem with the affordance-based approach to reproducer classification. Indeed, it may be useful to shift the line between trivial and non-trivial reproduction depending on the application, e.g., some computer viruses may be so reliant on external help that they are minimally autonomous and therefore are easily detected at run-time — it would be natural therefore to classify these reproducers as trivial.

A potential practical application of model selection is in the field of computer anti-virus scanning technology, where it may be possible only to scan certain interactions between running programs and the computer system that they run on. For instance, if we were to monitor hardware interrupts only (e.g., disk input/output routines), we would be in a situation analogous to Model M , where the actions afforded by the file store (cf. hard drive) are the only potential sources of information on viral activity at run-time. If we were to monitor both the hardware interrupts and the OS function calls, then we are in a situa-

tion analogous to Model N where all OS function use (including calls to read from or write to the file store) is afforded by external agents to the copier virus, and consequently more actions are available for the purposes of scanning for viral behaviour at run-time. This increase in dependence on external agency is reflected in the categorisation for the copier computer virus: Type III in Model M , and Type IV in Model N .

3.3 Worked Example: Langton's Loop

Langton's loop [15] reproduces on a two-dimensional cellular automaton grid. It consists of an outer "sheath" which contains the self-description: a series of symbols encoded in the states of the sheathed cells. The self-description causes an "arm" to be extended from one corner of the loop, which then turns perpendicularly, before repeating the process until a child loop has been constructed completely.

There are a number of ways in which the reproductive process of the loop could be modelled. For example, we could model the reproductive process so that S would be the set of states of the cellular automaton during the reproductive process. However, it is clearer and more succinct to use the more abstract model which follows.

Let S , A and \mapsto be defined so that $s_1 \xrightarrow{sd} s_1$ and $s_1 \xrightarrow{rm} s_2$. This follows from the definition of the loop, which has a self-description (sd) in the initial state, and therefore this step of the reproductive process is achieved in s_1 , without a state transition. The reproductive mechanism, the means by which the self-description is used as instructions to create an offspring, is achieved over a period from s_1 to s_2 . This transition corresponds to the states of the cellular automaton during the loop's reproductive process. Let $Ent = \{L\}$ where L is the Langton's loop, and $Aff(L, L) = \{sd, rm\}$. Therefore, L is a Type I reproducer.

An alternative classification is possible if we consider the transition rule of the cellular automaton to be an external entity which affords all state transitions to any other entity, including the loop. Therefore, both the self-description and reproductive mechanism actions are afforded by the transition rule, i.e., $sd, rm \in Aff(tr, L)$ where tr is the transition rule of the CA. Since all reproductive actions of the loop are now afforded by another entity (tr), we know that in this model the loop is a Type IV reproducer.

3.4 Worked Example: Bacteriophage Viruses

Bacteriophage viruses [22] infect bacteria in order to reproduce. The life cycle is highly complex, and to model it at the chemical level would be im-

practical. However, the cycle can be described abstractly as follows. Before reproduction, the bacteriophage virus and bacteria are immersed in an aqueous solution. When the virus encounters a bacterium the virus attaches itself to the bacterium. Penetration of the cell by the virus then occurs. This process introduces the virus's self-description (encoded in DNA or RNA) to the host cell. Next, the virus will hijack the host cell's reproductive mechanism through the infected self-description. Copies of components of the virus are then synthesised by the bacterium's own reproductive mechanism. These components are then assembled during a process called maturation, and finally, the viruses are released in process known as lysis. Lysis results in the death of the host cell.

Therefore we can identify a six-action reproductive process:

$$s_1 \xrightarrow{a} s_2 \xrightarrow{p} s_3 \xrightarrow{i} s_4 \xrightarrow{s} s_5 \xrightarrow{m} s_6 \xrightarrow{r} s_7$$

where a , p , i , s , m and r correspond to attachment, penetration, introduction (of the self-description), synthesis, maturation and release respectively. The action sd corresponds to the self-description, which is built-in. Therefore, S_v , A_v and \mapsto_v are defined implicitly. It would seem natural, from the viewpoint of the field of virology, that there should be two entities v and h corresponding to the virus and host cell respectively, and therefore $Ent_v = \{v, h\}$. We can determine the composition of the affordance sets by the analyses given in the literature (see, e.g., [22, 12]). The attachment phase is afforded by the virus, and in some cases, the host cell which may have receptors onto which the virus may attach itself. Therefore $a \in Aff(h, v)$ and $a \in Aff(v, v)$. The penetration and introduction of viral DNA/RNA actions are performed by the virus alone; the host cell does not assist, and therefore $p, i \in Aff(v, v)$. The synthesis and maturation actions are afforded by the host cell to the virus, since it is the host cell's biosynthetic machinery that enables these actions, and so $s, m \in Aff(h, v)$. The release stage is afforded to the virus by the virus and the host cell, since lysis is caused by proteins produced by the cell but which are coded for in the viral DNA/RNA. Therefore $r \in Aff(h, v)$ and $r \in Aff(v, v)$. Since the virus is reproducing it must be present in each stage of its reproductive cycle, and therefore $\forall s \in S : v \varepsilon s$. Additionally, since the host cell assists in actions a , s , m and r then we know that $\forall s \in \{s_1, s_4, s_5, s_6\} : h \varepsilon s$. So, in order to classify the bacteriophage virus we must determine which of the actions corresponds to the self-description and reproductive mechanism (abstract actions sd and rm respectively).

The virus contains its self-description in the form of its DNA/RNA. This genome reaches the necessary point for the reproductive mechanism by the process of penetration and introduction. Therefore p and i correspond to sd .

The reproductive mechanism of the virus is the means by which the self-description can be interpreted as a set of instructions in order to complete the reproductive process. Since the actions $\{a, s, m, r\}$ are vital to this goal, they must correspond to the reproductive mechanism rm . Since $p, i \in \text{Aff}(v, v)$ and $p, i \notin \text{Aff}(h, v)$, we know that the bacteriophage virus is either Type II or Type I. However, since $r \in \text{Aff}(h, v)$, which corresponds to part of the reproductive mechanism, we know that the virus must therefore be Type II.

3.5 Refinements of Reproduction Modelling

3.5.1 The Modified Type I Conjecture

Reproducers that lie outside Type I necessarily lack a completely self-afforded self-description or reproductive mechanism. However, in order to complete the reproductive process they must obtain, via external agency, the reproductive mechanism and self-description which they lack. Therefore, it follows that at some point during the reproductive process, any reproducer outside Type I must form a complex which is Type I with respect to the reproducer. In previous work [27] this was referred to as the Type I conjecture.

For example, when the T4 bacteriophage injects its viral DNA into the host bacterium, the resulting T4–bacterium complex is Type I with respect to the T4 bacteriophage, in that the self-description and reproductive mechanism are completely afforded by the complex to itself. For the worked example in 3.2, the copier virus forms a Type I reproducer when it uses the file store or operating system to obtain a sufficient reproductive mechanism and self-description. Gliders in Conway’s Game of Life [9] become Type I when the cellular automaton transition rule, which is a reproductive mechanism, is combined with the instant state of the CA, which serves as a self-description. The modified von Neumann reproducing automata in Types II or III (see 2.3, 2.4) must form a complex by external agency which is of Type I, in order to complete their reproductive processes.

It would be difficult to prove this phrasing of the Type I conjecture, since it would involve analysis of all reproducers outside Type I. However, our formal definition of reproduction models lets us phrase conjectures that would be proveable, and to this end we present another version of the Type I conjecture: “For all reproduction models outside Type I, there exists another model that represents the same reproductive process, but in which the entities and resulting affordances are modified so that the reproducer is Type I.”

The modified Type I conjecture is restricted to models of reproduction models, but given that reproduction models are adequate representations of

the reality of reproductive systems, no generality is lost. Despite the differences in phraseology, the meanings of the two conjectures are similar. For example, in the case of a T4 bacteriophage that forms a Type I complex after infection of a host cell, we might build a “natural” model that places the T4 in Type II. However, we can construct a model in which T4 is Type I, if we *aggrandise* the T4 and host cell entities so that they become a single entity, T4', and therefore every action afforded by the host cell to T4 is now afforded by T4' to itself. Since the host cell was the only external agent, which forced T4 to become categorised as Type II, in the new model there are no external agents, and every reproductive action is afforded by T4' to itself. Therefore T4' is Type I in this new model.

This aggrandisement process is analogous to the process by which T4 becomes a Type I complex in the original conjecture, except that now this Type I conversion takes place by construction of a new model, rather than as a side-effect of the reproductive process itself. Details of the proof of this conjecture will appear in [25].

3.5.2 Deities and the Type IV Conjecture

As we saw in the classification of Langton's loop, it is possible to introduce an entity into a reproduction model that effectively assumes responsibility for every action that takes place in the model. Therefore, every action is then afforded by this new “god-like” entity, and the reproducer becomes categorised as Type IV. Typically, the formation of such an entity happens when we take some implicit functionality (such as the transition rule for Langton's loop, or the laws of nature for a T4 bacteriophage virus, or the processor of computation for a computer virus) and give it to an entity which previously did not exist due to the ontological decisions made in construction of the reproduction model. The ultimate expression of such an entity would be some “deity”, without whom no actions can happen, and therefore whose presence implies that all reproducers in reproduction models which incorporate the deity as an entity must be Type IV. The possibility of categorisation of any non-Type IV reproducer as Type IV by introduction of a deity therefore becomes the Type IV conjecture.

3.5.3 Dealing With Evolution: Self-Sets

So far we have formalised reproduction using a reproduction model in the form of a tuple, one member being some $r \in Ent$ which represents the entity which reproduces. For some reproductive process $s_1 \mapsto s'_1 \mapsto \dots \mapsto s_2$ it is

intuitive to assume that $r \in s_1$ and $r \in s_2$, that is, the reproducer is present in both the start and end states. This is a direct result of our assertion that r is a reproducer. What happens, however, when the offspring of r is different from r ? In this case, we might say that the new r (which we shall call r') is present at the end of the process, i.e., $r' \in s_2$. However, r' is a reproducer, and when it comes to reproduce the offspring of r' (which we shall call r'') might differ from both r and r' . Here, r , r' and r'' are all entities, but only r appears in the reproduction model.

In order to avoid being forced to enumerate all the possible offspring of some reproducer r within the set of entities Ent , it seems reasonable to assert that $r \in Ent$ is some abstraction of the reproducer r , and that any of a number of versions (descendants) of the reproducer correspond to r . We will express this notion using *self sets*.

A self set is defined with respect to a given reproducer, under the condition that every element in the set is produced by some other element in the set by an act of reproduction. A self set therefore contains all possible descendants (and ascendants) of a reproducer, as well as the reproducer itself. Self sets are similar to Cohen's *viral sets* [5], which are constructed with the condition that a viral set contains only computer viruses that are generated by another virus in the set as a result of computation. Therefore, self sets can be seen as a generalisation of viral sets to the class of reproducers. We replace "computation" in Cohen's definition by "reproduction", which can itself be a computational process, but this is not a strict requirement.

4 Conclusion

We have shown that the "classic" examples of reproduction (including biological organisms and von Neumann reproducing automata) can be classified within Type I. We have defined two further Types (II and III) of reproducers that lack some critical part of their reproductive machinery, and a fourth Type (IV) which has no completely self-afforded reproductive machinery, but reproduces trivially thanks to external agency (e.g., cellular automaton gliders and the photocopy). This provides a means of separating trivial reproducers (i.e., those in Type IV) from non-trivial reproducers using the theory of affordances to show the dependence on external agency. The problem of separating trivial reproducers from non-trivial reproducers was highlighted by Langton [15], and the work here suggests that the reproducers intuitively thought of as being trivial are those found in Type IV. Langton says that for non-trivial reproducers,

“... responsibility for the production of the offspring should reside primarily within the sequences of actions undertaken by the parent structure. Note that we want to require that responsibility reside *primarily* with the parent structure itself, but not *totally*.” [15]

By using the affordance-based theory of reproduction models (see, e.g., 3) we have begun to state, qualitatively and quantitatively, in what ways the “responsibility” for the production of offspring is divided between the parent (reproducer) and the environment (other entities), and what the difference is between “primarily” (e.g., non-trivial reproducers outside Type I) and “totally” (Type I).

We gave an example of how a computer virus can be analysed to discover its reproductive reliance on external agents such as the file store or operating system, in terms of acquisition of a self-description and its reproductive mechanism. Such analyses of computer viruses could be a practical application in computer virology and anti-virus software engineering, as the less help a virus needs to reproduce, the more autonomous it is, and the less it relies on external agents (e.g., the operating system), thus making it more difficult to detect at run-time. For more information, see Section 4.2.3.

4.1 Comparison with Other Approaches

Reproducer classification based on reliance of a reproducer on the environment can also be found in the works of Freitas and Merkle (ch. 5, [8]), Taylor [19] and Luksha [16]. Freitas and Merkle give categories for the location of replication information (i.e., self-description) and replicator parasiticity (i.e., reliance on external agency for the reproductive machinery). Taylor divides the reproducer space into two: reproduction occurs either with or without reliance on external agency (auto- and assisted-reproduction respectively). Our approach differs from the aforementioned in that it allows a variety of possibilities between full reliance and non-reliance on external agencies for the reproductive mechanism and self-description (see 2.6). For example, within our framework it is possible that the set of actions corresponding to the reproductive machinery and/or the self-description of a reproducer is afforded partly by an external agent, and partly by the reproducer itself. A more finely-grained categorisation of reproducers along either axis is therefore possible.

Luksha offers a categorisation of reproducers based on the relative complexities of the reproducer and its environment. In our approach categorisation is based on the amount of reliance on external agency for the self-description

and reproductive machinery, which we identify as two crucial criteria for reproducer classification.

4.2 Future Work

4.2.1 Metrics for Reliance on External Agency

In 2.6 we described how reproducers outside Type I can be subclassified according to whether they afford themselves part of their reproductive mechanism and/or self-description (i.e., active) or not (i.e., passive). However there may be further opportunities to sub-classify based on other factors. For example, if we see the act of reproduction as a computational process of a certain minimal complexity, then if the actions that a reproducer affords itself together are the complexity of the whole reproductive process, then there must necessarily be some other (external) entity that compensates for this. Therefore, when comparing two reproducers in a similar environment (e.g., two computer viruses) that are both of a certain type outside Type I, then we can compare their reproductive reliance on external agency by comparing the complexity of those reproductive actions that the reproducer affords itself. The more complex the reproducer's self-afforded actions, the less the reliance on external agency. Of course, this presupposes the existence of some level of abstraction at which we can compare the complexity of actions, but in several cases, such as computer viruses, Tierran organisms, cellular automaton reproducers, etc., such a comparison would seem possible. Different methods of complexity could be used, e.g., space/time complexity, or the Kolmogorov complexity of the reproducer itself.

There is empirical evidence of differing degrees of reliance on external agency with respect to biological viruses. It is known that, “[viruses] with large genomes depend less on host functions than those with small genomes” [12]. In terms of our ontology, this states that the information content in the self-description (genome) is related to the reliance on external agency (the host cell). Another possible extension of this work would be use the methods described above to formalise this statement within our ontology.

4.2.2 Strategies for Reproduction

In December 2000, a relatively unprolific virus on the Windows 32-bit platform was able to infect executable files containing prolific network worms [3]. The destructive payload of the virus combined with the infectiousness of the worms created dangerous hybrids that were not predicted by the vendors of

anti-malware software. These hybrids were an emergent property of a complex “ecology” of reproducers.

A useful extension of this work would be to be able to analyse ecologies of reproducers, i.e., systems where more than one reproducer is present. Such ecologies could be constructed using affordances common between entities, for example, a bacterium might afford a site of infection for a bacteriophage virus, without necessarily specifying which virus might infect the bacterium. The labelled transition systems of the different reproductive processes could be combined using techniques such as those developed in process algebra [1]. In real-life ecologies, reproducers are capable of interesting behaviours such as crossing a species gap (e.g. biological viruses), or spontaneous virus–worm hybridisation (see above). In being able to build models of ecologies of reproducers by combining their models in a formal way, we could begin to analyse and predict interesting emergent properties of multi-reproducer systems.

4.2.3 Computer Virology: Anti-Virus Applications

In computer virology, computer viruses and network worms spread within computer systems whilst anti-virus software scans for suspect behaviours typical of reproducing malware in a process known as dynamic analysis [7]. As discussed in Section 3.5, it is possible to categorise reproducers in a number of ways, from Type I, through Types II and III, to Type IV, depending on how we model the entities and affordances in those models. A practical application of this flexibility of classification is in prioritisation of dynamic analysis on systems where resources are limited, e.g., on PDAs, smartphones, PCs, or other pervasive computing applications. Dynamic analysis depends on the ability of anti-virus software to intercept communications between reproducing malware and external entities such as the operating system, daemons/services, the file-store, network protocols, etc. Malware typically must enlist the help of these other entities in its reproductive process. Antivirus software is able to analyse this behaviour and flag it as suspicious in order to detect files infected by malware. In order to apply our ontology, we can say that the act of dynamic analysis by the antivirus software places constraints on the reproduction model that we construct. For example, if the antivirus software is able to intercept calls by a computer virus to the file store (during disk input/output operations, for example), then it is logical to classify the virus and the file store as separate entities. If the anti-virus software cannot intercept calls to the operating system, then effectively it cannot “distinguish” between the virus and the OS, and within the reproduction model we should treat them as one entity. So, when malware is afforded an action by an external entity, and the anti-virus software

is able to detect this, the anti-virus scanner has a better chance of detecting the malware than if it could not detect this behaviour.

By classifying malware according to this schema, we will know that the most difficult malware to detect at run-time will be that classified as Type I, because the anti-virus software cannot detect the behaviour of these viruses and worms, because it cannot intercept the calls made by the virus to external entities. The viruses in Types II, III and IV will have behaviours that are detectable by the anti-virus software. So, on a system where resources are limited, the anti-virus analysis scanner can focus its static analysis attentions on the malware in Type I, because this cannot be detected at run-time, and should therefore be high priority.

References

- [1] Baeten. J., 2005, *A Brief History of Process Algebra*, Theoretical Computer Science, Vol.335, pp.131-146.
- [2] Bench-Capon T., Malcolm. G., 1999, *Formalising ontologies and their relations*, in Bench-Capon T., Soda G., Toa A. M. (Eds.), Proceedings of the 16th International Conference on Database and Expert Systems Applications (DEXA '99), Springer Lecture Notes in Computer Science, Vol.1677, pp.250-259. Springer, Berlin.
- [3] Symantec Press Centre, 2000, *Symantec warns computer users of destructive Christmas Day virus/worm mutation*, http://www.symantec.com/region/reg_ap/press/my_001219b.html (accessed 2007-05-19).
- [4] Cohen F., 1987, *Computer viruses — theory and experiments*, Computers and Security, Vol.6, No.1, pp.22-35.
- [5] Cohen F., 1989, *Computational aspects of computer viruses*, Computers and Security, Vol.8, pp.325-344.
- [6] Dawkins R., 1990, *The Selfish Gene*, Oxford University Press, USA, 1990, Ch.11, pp.189–201, first published 1976.
- [7] Filiol E., 2005, *Computer Viruses: from Theory to Applications*, Springer, pp.151-163.
- [8] Freitas R. A. Jr., Merkle R. C., 2004, *Kinematic Self-Replicating Machines*, Landes Bioscience.

- [9] Gardner M., 1970, *Mathematical games: The fantastic combinations of John Conway's new solitaire game 'life'*, Scientific American, Vol.223, pp.120-123.
- [10] Gibson J. J., 1977, *The theory of affordances*, Perceiving, Acting and Knowing: Toward an Ecological Psychology, pp.67-82.
- [11] Goguen J. A., Malcolm G., 1996, *Algebraic Semantics of Imperative Programs*, Massachusetts Institute of Technology.
- [12] Granoff A., Webster R. G. (Eds.), 1999, *Encyclopedia of Virology*, Academic Press, Vol.3, pp.1414-15.
- [13] Hofstadter D. R., 2000, *Gödel, Escher, Bach: an Eternal Golden Braid*, Penguin, Ch.16, p.499.
- [14] Jones M. D., *Tevenphage.png*, <http://en.wikipedia.org/wiki/Image:Tevenphage.png> (accessed 2006-10-28).
- [15] Langton C. G., 1984, *Self-reproduction in cellular automata*, Physica D: Nonlinear Phenomena, Vol.10, pp.135-144.
- [16] Luksha P. O., 2003, *The firm as a self-reproducing system*, Proceedings of 47th International System Science Society Conference.
- [17] Ray T., 1994, *Evolution, complexity, entropy, and artificial reality*, Physica D, Vol.75, pp.239-263.
- [18] Schrödinger E., *What is Life?*, Cambridge University Press, 1944.
- [19] Taylor T. J., 1999, *From Artificial Evolution to Artificial Life*, PhD thesis, University of Edinburgh.
- [20] Torenvliet G., 2003, *We can't afford it!: the devaluation of a usability term*, Interactions, Vol.10, No.4, pp.12-17.
- [21] von Neumann J., 1966, *Theory of Self-Reproducing Automata*, University of Illinois Press.
- [22] Voyles B. A., 2002, *The Biology of Viruses*, McGraw Hill.
- [23] Weaver N., Paxson V., Staniford S., Cunningham R., Spring 2006, *Life*, in E. N. Zalta (Ed.), The Stanford Encyclopedia of Philosophy <http://plato.stanford.edu/archives/spr2006/entries/life/> (accessed 2007-05-19).

- [24] Webster M., 2005, *Algebraic specification of computer viruses and their environments*, in Mosses P., Power J., Seisenberger M. (Eds.), Selected Papers from the First Conference on Algebra and Coalgebra in Computer Science Young Researchers Workshop (CALCO-jnr 2005), University of Wales Swansea Computer Science Report Series CSR 18-2005, pp.99-113.
- [25] Webster M., Malcolm G., *Classifying and relating models of reproducers using the theory of affordances*, in preparation.
- [26] Webster M., Malcolm G., 2006, *Detection of metamorphic computer viruses using algebraic specification*, Journal in Computer Virology, Vol.2, No.3, pp.149-161. DOI: 10.1007/s11416-006-0023-z.
- [27] Webster M., Malcolm G., 2007, *Reproducer classification using the theory of affordances*, Proceedings of the 2007 IEEE Symposium on Artificial Life (CI-ALife 2007), pp.115-122. IEEE Press.