

# Reproducer Classification Using the Theory of Affordances

Matt Webster

Department of Computer Science  
University of Liverpool  
Liverpool, L69 3BX, UK  
matt@csc.liv.ac.uk

Grant Malcolm

Department of Computer Science  
University of Liverpool  
Liverpool, L69 3BX, UK  
grant@csc.liv.ac.uk

**Abstract**—We present a new approach to the classification of reproducers based on an affordance theory of reproductive behaviour. First, we define the notion of an affordance as an action that one object in an environment can perform for another object. Using this ontology we can classify the reproducer space according to the presence (or absence) of a self-description and/or reproductive machinery. We give examples of how various reproducers (both natural and artificial) can be categorised, and show how this ontology can be used to separate trivial from non-trivial examples of reproduction. With a worked example we show how we might use this approach to classify computer viruses, and gain insight into their reproductive reliance on external agency. Finally, we conjecture that reproduction requires a self-description and a reproductive mechanism, whether it is supplied from within or from an external agent.

## I. INTRODUCTION

The ability to reproduce is a necessary condition in most definitions of what it means to be alive (see, e.g., [1], [2]). Defining life seems to be contentious and prone to admitting at least some false positives: examples such as biological viruses, that meet all the defining criteria, yet go against intuitions about what life actually is. In contrast, defining reproduction seems relatively straightforward. There are clear paradigmatic examples of reproducers: biological organisms [3], von Neumann’s self-reproducing automaton [4], computer viruses [5], and so forth. Despite its comparative straightforwardness, however, there are many examples of reproducers that stretch intuitive acceptance of what it means to be a reproducer. Such examples include fire, seeding crystals, photocopied documents, fixed points of functions, and even a pen on a desk, which reproduces itself from one instant to the next.

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 [6]. 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., [7]); a comprehensive overview of the various 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 reproduction. In Section II, 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 agents.

Section II also gives types for each class of reproducer, e.g., Von Neumann’s self-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 III 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 a worked example of a computer virus that relies on the agency of an operating system in order to obtain a self-description. Although the presentation of this model is merely sketched, it is based on a very precise formal model presented as an equational theory of the semantics of assembly language [9], [10]. This more formal approach allows us to formulate relationships between models, which means that our classification is much more like a structured space of reproducers. As an example, Section III concludes with a conjecture relating all models to what we call “Type I” models.

We believe that our classification provides useful insights with particular applications to computer virology. Although this paper presents only the first steps towards an ecological understanding of agents interacting through mutual affordance, directions for future research are discussed in the concluding section.

## II. CLASSIFICATION USING AFFORDANCES

Here we present four abstract types for reproduction, based on analysis of classic formal models of reproduction, e.g., von Neumann’s self-reproducing automaton in an unmodified and two modified versions to define the different types of non-trivial reproducers (*Types I, II and III*), and gliders such as those seen in Conway’s Game of Life [11] to define trivial reproducers (*Type IV*).

We shall see that these types are the four corners of a reproducer classification space, and that real-life reproducers (such as biological organisms, artificial organisms, and computer viruses) occupy different points within this space depending on their reproductive reliance on themselves and their environment.

### A. Affordance Theory

Affordance theory describes the functional relationships between an animal and its environment [6]. 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.

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.

(Please note: the word “affordance” has come to mean “perceived affordance” in the domain of human-computer interaction. Torenvliet [12] gives a thorough discussion of the confusion that has arisen, and the correct meaning of the word.)

1) *Philosophical Note:* Gibson describes a niche in an environment, into which an organism fits, as a “set of affordances” [6]. In this paper we categorise different types of reproducer according to the sets of affordances corresponding to different reproductive niches. Gibson’s affordance theory was preceded philosophically by Hume’s bundle theory of ontology [13], which sought to describe all objects in terms of their functions only. In that sense, we seek to gain insight into the different categories of reproducer through a bundle theory of reproduction.

### B. Type I

John von Neumann’s self-reproducing automaton was the first mathematical simulation 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 [4]. From an abstract point of view, this is the same process that biological organisms undergo during reproduction, where the genome (cf.

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.

1) *Example: Langton’s Loop:* Langton’s self-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 [14]. 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.

### C. Type II

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

1) *Example: Automaton with Self-Description:* An example of a Type II reproducer would be an automaton similar to the von Neumann self-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) *Example: T4 Bacteriophage:* The T4 bacteriophage affords itself a complete and 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

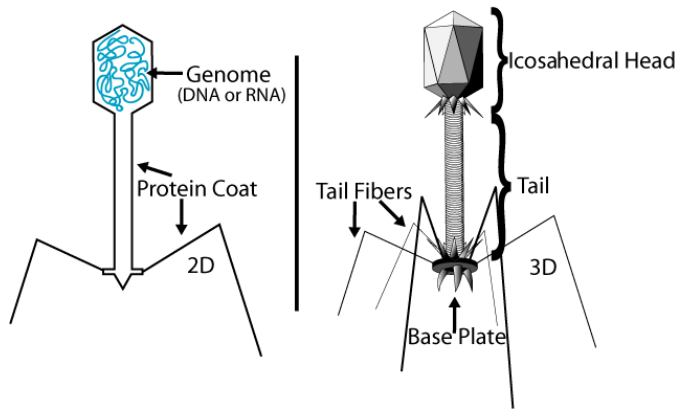


Fig. 1. The T4 bacteriophage virus. The legs of the T4 attach themselves to the cell wall of a bacterium. Then, using the base plate and tail as an injection mechanism, the genetic material in the head is transferred to the interior of the host cell, where it hijacks the reproductive mechanism of the cell in order to reproduce. Image source: [15].

the reproduction are afforded by the bacterium, we can say the mode of reproduction for the T4 bacteriophage is Type II.

#### D. Type III

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 self-reproducing automaton, we can visualise an Type III reproducer as being a von Neumann self-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 self-reproducing automaton.

1) *Example: Compiler:* A compiler (in self-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) *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.

#### E. Type IV

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.

1) *Example: Game of Life Gliders:* Cellular automaton (CA) gliders in Conway's Game of Life [11] 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) *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.

The photocopy as an example of trivial reproduction was given by Taylor [16].

#### F. Degrees of Reliance on External Agency

We have given four types of reproduction based on a reproducer's ability to afford itself completely either a reproductive mechanism or a self-description (see Fig. 2). Where a reproducer lacks ability in either way (i.e., in Types II, III and IV) it should be possible to further categorise reproducers according to degrees of reliance on external agency. For example, a T4 bacteriophage affords itself part of its reproductive mechanism (i.e., the DNA/RNA injection mechanism), and is therefore less reliant on external agency than, for example, a disabled T4 bacteriophage that cannot perform the DNA/RNA injection action.

### III. A FORMAL APPROACH TO MODELLING AFFORDANCES

The preceding discussion has focused on the distinguishing characteristics of the four types of reproduction. These char-

	Type III	Type I
$rm \in Aff(r, r)$	e.g., von Neumann automaton without tape	e.g., von Neumann automaton with tape
	Type IV	Type II
$rm \notin Aff(r, r)$	e.g., cellular automaton gliders	e.g., automaton with self-description
	$sd \notin Aff(r, r)$	$sd \in Aff(r, r)$

Fig. 2. The four reproducer types, where  $r$  is a reproducer,  $rm$  is the action corresponding to the reproductive mechanism,  $sd$  is the action corresponding to the self-description and  $Aff(r, r)$  is the set of actions that  $r$  affords itself. If some action  $a \notin Aff(r, r)$ , then  $a \in Aff(e, r)$ , where  $e$  is some external entity.

acteristics 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 self-reproductive processes, but to be able to compare and relate different models. This might allow us, for example, to identify strategies for self-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., [17]).

In this section, we give a more formal approach to modelling self-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 a worked example in Section III-B of a copier computer virus. Finally, Section III-C gives a first step towards relating models of reproductive processes by stating a conjecture that all such processes can be viewed at some stage in their reproduction processes, and at some level of abstraction, as Type I reproducers.

#### A. 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 self-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 III-B 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 [10], [9]. In more “real life” examples, such as the T4 bacteriophage, 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 may be several states that may 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 “ $r : s$ ” — 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  $r : s'$ . 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 self-reproducing automaton, and the succession of states represents all the intermediate states involved in the construction of its copy. We assume that the relation  $r : s$  can be made abstract enough to accommodate an appropriate laxity in the notion of entity: i.e., we should gloss  $r : s$  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 [5], 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.”

In the previous section, we saw that it was possible to characterise various types of self-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 model of a reproductive system consists of:

- a set  $S$  of states;
- a set  $Ent$  of entities;
- a relation  $_ : _$  between entities and states, where for entities  $e$  and states  $s$ ,  $e : s$  indicates that  $e$  is present in the state  $s$ ;
- a set  $A$  of actions;
- a ternary relation  $\mapsto$  of *succession*:  $s \xrightarrow{a} s'$  means that the action  $a$  occurring in the state  $s$  leads to the new 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  $e' : s$ :  $a$  is possible in  $s$  (i.e., if  $s \xrightarrow{a} s'$  for some state  $s'$ ) if and only if  $e : s$ . Notionally,  $Aff(e, e')$  is the set of affordances that  $e$  gives to  $e'$ .

As an immediate corollary of this definition, we have that an action  $a$  is afforded by some entity  $e$  to itself implies that the action  $a$  is possible in all states in which  $e$  is present. That is, the ability to perform that action depends only on the presence of  $e$  itself.

We saw above that this basic framework allows us to characterise reproduction, in a way that allowed trivial, i.e., Type IV, examples. We can specify Type I reproduction by requiring the following: some entity  $r$  (the reproducer); states  $s_0$ ,  $s_1$  and  $s_2$ , with  $r : s_0$  and  $r : s_2$ ; a sequence  $sd = a_1, \dots, a_n$  of actions (which notionally obtain a self-description of  $r$ ), with  $s_0 \xrightarrow{sd} s_1$ ; an entity  $d$  (the description) with  $d : s_1$ ; and a sequence  $rm = b_1, \dots, b_m$  of actions (which notionally is the process of creating a copy from the description) with  $s_1 \xrightarrow{rm} s_2$ . Finally, we require that this be Type I reproduction, by requiring that  $rm, sd \in Aff(r, r)$ . (Note that we freely extend the succession relation  $s \xrightarrow{a} s'$  and the affordance function  $Aff$  to sequences of actions; the details are quite straightforward, and omitted here.) Specifications of Types II, III and IV are similar.

### B. Worked Example

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 [10]. The syntax and semantics of SPL are defined formally using OBJ, a formal notation for algebraic specification [18].

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 lines 6–10 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 whose handle is `nfh` using the `writeToFile(_, _)` function. In line 8 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 13) 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 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 12 the temporary file `nfh` is deleted from the file system. Line 13 is the label mentioned in the previous paragraph, and the final line (14) denotes the end of the file.

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.

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

```

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

```

Fig. 3. Copier computer virus programmed in SPL.

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  $\Psi$  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 [10].) Since  $\Psi$  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  $\llbracket \psi_x \rrbracket(s) = s'$  where  $\llbracket \psi_x \rrbracket(s)$  is the effect of  $\psi_x$  on store  $s$ , which is to say that a state transition from  $s$  to  $s'$  under action  $\psi_x$  is only possible if the effect of  $\psi_x$  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  $N$ .

```

 $\psi_1 = fh1 := getFileHandle$ 
 $\psi_2 = myName := getSelfName$ 
 $\psi_3 = nfh := newFileHandle$ 
 $\psi_4 = counter := 0$ 
 $\psi_5 = lastLine := label end$ 
 $\psi_6 = do \{ \dots \} while ( \dots )$ 
 $\psi_7 = line := getLine(myName,counter)$ 
 $\psi_8 = writeToFile(line,nfh)$ 
 $\psi_9 = counter := s(counter)$ 
 $\psi_{10} = prepend(nfh,fh1)$ 
 $\psi_{11} = deleteFile(nfh)$ 
 $\psi_{12} = label end$ 

```

Fig. 4. Numbering of the SPL statements from the copier computer virus.

2) *Classifying the Copier Virus Using Model M*: Model  $M$  has two entities: the copier virus and the file store. Therefore,  $Ent_M = \{cv, fs\}$ . Since we have two entities, there are two affordance sets  $Aff(cv, cv)$  and  $Aff(fs, cv)$  we must consider. (Since we are trying to assess the contribution of external agency to  $cv$ , it is unnecessary to calculate the sets  $Aff(cv, fs)$  and  $Aff(fs, fs)$ .)

Numbering the statements from  $\psi_1$  to  $\psi_{12}$  (see Fig. 4) we can calculate the dependence on the operating system using set analysis. Let  $rm$  be the set of actions facilitating the reproductive mechanism, and let  $sd$  be the set of actions facilitating the self-description acquisition mechanism. Then,  $sd = \{\psi_2\}$  since this is the statement used to obtain a file handle for acquisition of the self-description (i.e., SPL code) and  $rm = \{\psi_8, \psi_{10}, \psi_{11}\}$ , 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.,  $Aff(fs, cv) = \{\psi_1, \psi_2, \psi_3\}$ . The statements  $\psi_{4-12} \in Aff(cv, cv)$  since these actions only need the presence of the copier virus ( $cv$ ). (Even though some of  $\psi_{4-12}$  rely on the file handle obtained by  $\psi_2$  as an affordance from the file store, this information does not need to be retrieved again, and therefore the copier virus is able to modify files using the file handles directly and without the help of the file store.) Now we can calculate  $rm \cap Aff(fs, cv)$  and  $sd \cap Aff(fs, cv)$  in order to specify the reliance on external agency (i.e., in this case, the file store) by the copier virus. So,  $rm \cap Aff(fs, cv) = \emptyset$  and  $sd \cap Aff(fs, cv) = \{\psi_2\}$ . In fact,  $rm \subseteq Aff(cv, cv)$  and  $sd \subseteq Aff(fs, cv)$ , and 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 completely self-afforded (i.e.,  $rm \subseteq Aff(cv, cv)$ ) and the self-description is afforded (at least partially) by external agency (i.e.,  $sd \not\subseteq Aff(cv, cv)$ ).

3) *Classifying the Copier Virus Using Model N*: Model  $N$  has two entities: the copier virus and the operating system. Therefore  $Ent_N = \{cv, os\}$ .

Since we have two entities, there are two affordance sets  $Aff(cv, cv)$  and  $Aff(os, cv)$  we must consider. (As is the case

with Model  $M$ , we are trying to assess the contribution of external agency to  $cv$ , so it is unnecessary to calculate the sets  $Aff(cv, os)$  and  $Aff(os, os)$ .

Numbering the statements from  $\psi_1$  to  $\psi_{12}$  as before (see Fig. 4) we can calculate the dependence on the operating system using set analysis. Again, let  $rm$  be the set of actions facilitating the reproductive mechanism, and let  $sd$  be the set of actions facilitating the self-description acquisition mechanism. Then,  $sd = \{\psi_2\}$  and  $rm = \{\psi_8, \psi_{10}, \psi_{11}\}$ , 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.,  $Aff(os, cv) = \{\psi_1, \psi_2, \psi_3, \psi_7, \psi_8, \psi_{10}, \psi_{11}\}$ . The statements  $\psi_4, \psi_5, \psi_6, \psi_9, \psi_{12} \in Aff(cv, cv)$  since these actions only need the presence of the copier virus ( $cv$ ), as they do not use any OS functions. Now we can calculate  $rm \cap Aff(os, cv)$  and  $sd \cap Aff(os, cv)$  in order to specify the reliance on external agency (i.e., in this case, the operating system) by the copier virus. So,  $rm \cap Aff(os, cv) = \{\psi_8, \psi_{10}, \psi_{11}\}$  and  $sd \cap Aff(os, cv) = \{\psi_2\}$ . In fact,  $rm \subseteq Aff(os, cv)$  and  $sd \subseteq Aff(os, cv)$ , and therefore the copier virus is an 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 (i.e.,  $sd, rm \not\subseteq Aff(cv, cv)$ ).

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 model choice, 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 situation 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  $N$ .

### C. Type I Conjecture

Reproducers that lie outside Type I necessarily lack a complete 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.

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 III-B, 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 the Game of Life become Type I when the CA 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 self-reproducing automata in Types II or III (see II-C, II-D) must form a complex by external agency which is of Type I, in order to complete their reproductive processes.

## IV. CONCLUSION

We have shown that the “classic” examples of reproduction (including biological organisms and von Neumann self-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 [14], 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*.” [14]

By using the affordance-based theory of reproductive models (see, e.g., III-B) 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.

### A. 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 [16] and Luksha [7]. 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 II-F). 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.

### B. Future Work

1) *Metrics for Reliance on External Agency:* In II-F we described how reproducers outside Type I might be classified according to a sliding scale of reliance on external agency, from full dependence on external agency (e.g., a photocopy or a CA glider), through partial dependence (e.g., a T4 bacteriophage that affords itself only part of its reproduction mechanism), to zero dependence (i.e., Type I reproducers). It should therefore be possible to develop metrics for quantifying this variable dependence of reproducers on other entities. It may be possible to construct different metrics for the classes of non-trivially reproducing cellular automata, or artificial life

organisms such as those seen in Ray’s Tierra [19], for example. It is also possible to imagine metrics for use in the analysis and classification of biological viruses, if a standard and abstract means of defining and attributing the affordances of biological viruses and their hosts could be found.

2) *Abstract Reproductive Niches:* Another extension of this work (following on from III-B) would be to create a standardised set of reproductive affordances (an abstract reproductive niche) for the reproduction of computer viruses, and to use this to form a more reliable metric for the reliance of computer viruses and network worms on external agents such as operating systems, compilers, network data transfer protocols, etc.. Such a metric would provide a reliable means of quantifying viral autonomy, which could be of practical importance to the developers of anti-virus software.

### REFERENCES

- [1] B. Weber, “Life,” in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., Spring 2006, <http://plato.stanford.edu/archives/spr2006/entries/life/> (accessed 2006-10-28).
- [2] E. Schrödinger, *What is Life?* Cambridge University Press, 1944.
- [3] R. Dawkins, *The Selfish Gene*. Oxford University Press, USA, 1990, ch. 11, pp. 189–201, first published 1976. ISBN: 0192860925.
- [4] J. von Neumann, *Theory of Self-Reproducing Automata*, A. W. Burks, Ed. University of Illinois Press, 1966.
- [5] F. Cohen, “Computer viruses – theory and experiments,” *Computers and Security*, vol. 6, no. 1, pp. 22–35, 1987.
- [6] J. J. Gibson, “The theory of affordances,” *Perceiving, Acting and Knowing: Toward an Ecological Psychology*, pp. 67–82, 1977.
- [7] P. O. Luksha, “The firm as a self-reproducing system,” in *Proceedings of 47th International System Science Society Conference*, 2003.
- [8] R. A. Freitas Jr. and R. C. Merkle, *Kinematic Self-Replicating Machines*. Landes Bioscience, 2004, ISBN 1570596905. <http://www.molecularassembler.com/KSRM.htm> (accessed 2006-10-28).
- [9] M. Webster and G. Malcolm, “Detection of metamorphic computer viruses using algebraic specification,” *Journal in Computer Virology*, vol. 2, no. 3, pp. 149–161, December 2006, DOI: 10.1007/s11416-006-0023-z.
- [10] M. Webster, “Algebraic specification of computer viruses and their environments,” in *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, P. Mosses, J. Power, and M. Seisenberger, Eds., 2005, pp. 99–113, <http://www.csc.liv.ac.uk/~matt/> (accessed 2006-10-28).
- [11] M. Gardner, “Mathematical games: The fantastic combinations of John Conway’s new solitaire game ‘life’,” *Scientific American*, vol. 223, pp. 120–123, 1970.
- [12] G. Torenlvliet, “We can’t afford it!: the devaluation of a usability term,” *Interactions*, vol. 10, no. 4, pp. 12–17, July–August 2003.
- [13] D. Hume, *A Treatise of Human Nature*, 1740, book I, part I, sect. IV.
- [14] C. G. Langton, “Self-reproduction in cellular automata,” *Physica D: Nonlinear Phenomena*, vol. 10, pp. 135–144, 1984.
- [15] M. D. Jones, “Tevenphage.png,” <http://en.wikipedia.org/wiki/Image:Tevenphage.png> (accessed 2006-10-28).
- [16] T. J. Taylor, “From artificial evolution to artificial life,” Ph.D. dissertation, University of Edinburgh, 1999, <http://www.tim-taylor.com/papers/thesis/index.html> (accessed 2006-10-28).
- [17] T. Bench-Capon and G. Malcolm, “Formalising ontologies and their relations,” in *Proceedings of the 16th International Conference on Database and Expert Systems Applications (DEXA ’99)*, ser. Springer Lecture Notes in Computer Science, T. Bench-Capon, G. Soda, and A. M. Toa, Eds., vol. 1677. Springer, Berlin, 1999, pp. 250–259.
- [18] J. A. Goguen and G. Malcolm, *Algebraic Semantics of Imperative Programs*. Massachusetts Institute of Technology, 1996, ISBN 026207172X.
- [19] T. S. Ray, “An approach to the synthesis of life,” in *Artificial Life II*. Addison-Wesley, California, 1991, pp. 371–408, ISBN 0201525712.