A Semântica de Ações de Vírus de Computador

Lin Tse Min
ltm@di.ufpe.br

Silvio Lemos Meira
srlm@di.ufpe.br

Departamento de Informática
Universidade Federal de Pernambuco
Cidade Universitária, Caixa Postal 7851
50.732-970 - Recife/PE

Resumo

A Semântica de Ações é uma forma de Semântica Denotacional que tem como base o método algébrico para especificar Tipos Abstratos de Dados, desenvolvida com a finalidade de tornar as descrições formais mais compreensíveis e portanto, mais suscetíveis à utilização prática.

Os vírus de computador surgiram na década de 80 e desde então, têm infligido perdas a milhões de usuários em todo o mundo.

Neste trabalho é apresentada uma semântica formal de uma das mais disseminadas classes de vírus de computador, *Sexta-feira 13*.

Abstract

Action Semantics is related to Denotational Semantics, but it is based on the algebraic method for specifying Abstract Data Types. It was developed to make formal descriptions more understandable and so, easier to be used.

The computer viruses appeared in large scale in the 80's. Since then, users all over the world have been disturbed by different classes of viruses.

The formal semantics of a class of computer viruses is described here. It is given in action semantics style, showing some bugs in the viruses' implementation. The class analysed is the well-known "Friday The 13th".

1 Introdução

As descrições formais têm ganho "popularidade" como modelo de transmissão de informação de modo claro e não ambíguo. Dentre os vários estilos adotados para a definição formal de semânticas, alguns se destacam: Semântica Operacional [Plotkin81]; Algébrica [Guessarian81, Bergstra89], Axiomática [Hoare71, Goguen84] e Denotacional [Scott82, Schmidt86, Stoy86, Gunter89]. Estes formalismos, embora utilizados com sucesso em estudos teóricos, não se mostraram adequados para a descrição de linguagens de programação efetivas, como Pascal e ADA - as descrições obtidas eram incompletas ou grandes demais e quase incompreensíveis devido principalmente à falta de estrutura de tais métodos.

O enfoque mais promissor é o seguido pela semântica denotacional, que utiliza o conceito de "denotação", ou seja, funções semânticas mapeiam cada entidade sintática diretamente no seu significado (denotação), que é um valor matemático, como um número ou uma função que existem independentemente.

A semântica de ações [Mosses88, Mosses89, Mosses89a, Mosses89b] foi desenvolvida com o objetivo de tornar possível a especificação da semântica de linguagens de programação, minorando os problemas apresentados com os formalismos acima descritos. Para isto, aspectos pragmáticos como legibilidade, modularidade e reusabilidade foram considerados. Contudo, a utilização da semântica de ações não é restrita ao principal objetivo de sua concepção, prestando-se também para a descrição formal de conceitos de outras áreas da Ciência da Computação, como demonstrado em [Musicante90].

Este trabalho foi motivado por esta última corrente de pesquisa em descrições formais e objetiva a descrição do comportamento de um(a classe de) vírus de computador. A descrição via semântica de ações permite que o combate a tais programas seja efetuado de maneira mais clara e segura. Devido à abordagem rigorosa do método, a semântica é *clara* e *não ambígua*, simplificando o entendimento dos vírus e *desmistificação* que o combate efetivo aos mesmos, normalmente suposto estar ao alcance apenas de "superprogramadores" ou "gurus", apesar do caso "geral" tratado aqui ser apenas para o sistema DOS/80x86.

Na seção seguinte, o método utilizado para a descrição formal é apresentado. Uma introdução genérica aos vírus é apresentada na seção 3. Na seção 4 é descrita a semântica de ações de uma das mutações de vírus sexta-feira 13 e finalmente, a seção 5 contém as conclusões e direções futuras.

2 A Semântica de Ações

A semântica de ações foi desenvolvida por Peter D. Mosses na segunda metade da década de 80, utilizando as idéias de semântica denotacional e o método algébrico para especificar Tipos Abstratos de Dados [Burstall77, Goguen78, Goguen84]. Distintamente da semântica denotacional, onde as denotações são funções de alta ordem, a semântica de ações utiliza ações especificadas de forma algébrica como denotações. Ainda, a semântica denotacional tem como modelo apenas a notação λ (e suas variantes), enquanto a semântica de ações possui uma notação padrão razoavelmente grande para expressar ações.

As ações refletem diretamente a semântica operacional dos conceitos básicos do ambiente explorado, ou seja, sua essência computacional e não matemática. Ao ser "executada", uma ação utiliza dados (informações) passados a ela por ações anteriores e transmite dados para a próxima ação. As informações são processadas de forma gradual (e não instantaneamente). Uma ação pode apresentar os seguintes comportamentos:

- *Completar*, ou seja, terminar normalmente e passar informações para as ações subseqüentes;
- *Falhar*, quando termina de forma anormal (neste caso uma ação alternativa pode ser executada; se não existem ações alternativas, todas as ações subseqüentes também falham);
- *Divergir*, não terminando a execução (neste caso nenhuma outra ação subseqüente é executada). Uma ação que diverge é utilizada para representar a semântica de programas que não terminam (não necessariamente através de erros, como no caso dos sistemas operacionais);
- *Escapar*, ou seja, ter uma terminação com exceção (todas as ações subseqüentes não são executadas até que se encontre uma ação que trate a exceção). Uma ação que "escapa" é necessária para evitar a execução das ações subseqüentes que seriam executadas normalmente, permitindo assim que a próxima ação executada seja uma ação que trate da exceção.

Por outro lado, informações propagadas entre ações podem ser classificadas como:

- *Transientes* - são informações (produzidas por uma ação) que são passadas apenas no início da próxima ação. Uma informação transiente não é repassada adiante a menos que seja explicitamente "reproduzida", podendo ser utilizada para representar valores intermediários em uma computação;
- *Escopos* - são informações (produzidas por uma ação) que são passadas para a próxima ação, podendo ser modificadas. São utilizadas para representar ligações de valores a variáveis temporárias;
- *Estáveis* - são informações (produzidas por uma ação) que são passadas para todas as ações subseqüentes, podem ser modificadas por uma outra ação e são utilizadas para representar a atribuição de valores a variáveis;
- *Permanentes* - são informações com a mesma característica de informações estáveis, mas não podem ser modificadas, sendo utilizadas para representar a história de comunicação entre processos concorrentes.

Ações podem ser compostas através de "combinadores de ações". Um combinador de ação combina uma ou duas ações em uma ação mais complexa, determinando a ordem na qual as subações serão executadas, além de controlar o fluxo de informação entre essas subações. Dentre outros, são combinadores básicos de ações:

- [_or_] A ação "A1 or A2" escolhe uma ação entre A1 ou A2 a ser executada. Se a subação escolhida falhar, a outra subação é executada. A escolha da subação a ser executada é não-determinística.
- [_and_] A ação "A1 and A2" faz com que as subações A1 e A2 sejam executadas colateralmente, ou seja, as subações podem ser executadas em qualquer ordem ou mesmo intercaladas.
- [_and then_] A ação "A1 and then A2" faz com que as subações A1 e A2 sejam executadas sequencialmente. A2 é executada apenas se A1 completar.

Com os combinadores [_or_] e [_and then_], uma ação composta *completa* somente se ambas subações *completarem*.

A notação de ações fornece controle sobre o fluxo de informações e controle, permitindo assim a especificação de construtores tais como: expressão, comando e declaração. Em um determinado momento, ao se considerar um único tipo de fluxo de informação, diferentes "facetas" das ações podem ser relacionadas:

- faceta de *controle* - ignora o processamento de informações, considerando apenas a forma pela qual as subações são executadas.
- faceta *funcional* - informações completadas por uma ação são passadas diretamente para as próximas ações. Tais informações são chamadas de transientes e devem ser utilizadas de imediato (ou explicitamente passadas adiante), sob risco de desaparecerem. Por exemplo, em [A1 then A2], as ações A1 e A2 são executadas seqüencialmente e na faceta funcional, todas as informações transientes geradas por A1 são passadas diretamente e apenas para A2. As informações geradas pela ação composta são aquelas geradas pela ação A2.

- faceta *declarativa* - as ações recebem e produzem ligações de valores a variáveis - ou escopo. Alguns combinadores criam ligações, enquanto outros combinam os recebidos com os produzidos. Em A1 then A2, A1 e A2 recebem as mesmas ligações, que são combinadas e, se houver colisão, a ação composta falha. As informações geradas pela ação composta são aquelas geradas pelas subações.

- faceta *imperativa* - as ações têm acesso a um número arbitrário de células de memória. Cada célula pode conter um dado, estar indefinida (alocada mas sem conteúdo) ou ainda pode não estar alocada. O estado de cada célula pode ser alterado a qualquer momento por uma ação. Esta faceta está relacionada ao processamento de informações estáveis. Em A1 then A2, A1 e A2 são executadas seqüencialmente e as alterações realizadas por A1 precedem aquelas realizadas por A2, que sofre influências das alterações realizadas por A1 na memória.

- faceta *comunicativa* - ações recebem e enviam informações, que não podem ser alteradas por ações subseqüentes. Ações podem também subcontratar (através da ação **subcontract**) outras ações. Esta faceta corresponde ao processamento de informações permanentes.

3 Os Vírus de Computador

Os vírus de computador [Minasi91, Mismetti90, Ruber91, Weber89, Rubenking89, Seymour88, Fainberg89, Greenberg89] podem ser definidos como programas que têm a capacidade de infectar outros programas, modificando-os de forma a permitir a inclusão de sua cópia. Quanto à forma de ação, os vírus podem ser classificados como:

- "Inofensivos" ou "Brincalhões" - apenas contaminam, ou seja, ao se propagarem não causam nenhum dano físico (ao computador) ou lógico (aos programas e dados). No entanto, podem ocasionar diminuição do desempenho da máquina e emitir mensagens/sinais sonoros, além da redução da memória principal (RAM) ou espaço livre em disco.

- "Malignos" - além da contaminação, estes vírus podem estar relacionados a danos físicos (destruição ou desgaste premeditado do *hardware* do micro). A perda parcial ou total de informação/programas na memória principal ou em disco em geral se deve à atuação destes vírus.

Os vírus malignos mais prejudiciais são aqueles que não causam perda total de informações, mas apenas parcial ou aparentemente nula (através da troca de informações um vírus pode atuar sem ser facilmente detectado). Não ocasionando perda total, o vírus pode gradativamente, sem ser percebido, comprometer a tomada de decisões estratégicas (em uma empresa ou sistema de controle de vôo, por exemplo), uma vez que tais decisões podem ser fundamentadas em dados irreais. Há, por exemplo, um vírus conhecido como "Mailson" que, 30 minutos após o seu alojamento em memória, causa uma inversão das informações numéricas presentes no vídeo ou a serem emitidas para impressão.

Dentre os danos físicos que podem ser ocasionados por uma virose maligna, é interessante relacionar a gravidade de alguns:

- Um monitor pode ser queimado por sobreaquecimento, caso haja um chaveamento intensivo para um modo de vídeo não suportado pelo mesmo.

- Um acionador de disco flexível (disquete) pode também sofrer sobreaquecimento, caso o cabeçote seja mantido em "vai-vem" constante sobre a superfície do disco. Se constantemente acionado além dos limites normais das trilhas, um acionador pode ser desajustado (desalinhado).

- Alguns discos rígidos não podem sofrer formatação física, uma vez que contêm trilhas pré-formatadas pelo fabricante. A perda das informações gravadas nestas trilhas pode fazer com que a controladora do disco não o reconheça, ou o faça de forma errada. Discos que seguem o padrão IDE[1], adotado pela grande maioria dos micros do padrão IBM PC-AT (80286, 80386, 80486) e alguns do padrão IBM PC-XT (8088, 8086) estão sujeitos a este problema.

Ainda que mais de 1.302 [McAfee92] tipos (e mutações) de vírus já tenham sido detectados, a ntaminação pelos mesmos segue basicamente uma das duas formas descritas a seguir:

- Bloco de Carga (*BootStrap*)

  Nos microcomputadores, o sistema operacional (SO) não está localizado na ROM[2], mas sim em disco. Neste caso, a ROM contém algumas funções de baixo nível para as operações de entrada e saída (BIOS[3]). No final da rotina de inicialização do BIOS, um pequeno programa conhecido como programa de *bootstrap*, responsável pela carga do sistema operacional, é carregado do disco. Em geral estes programas de *bootstrap* são pequenos (menos de 512 bytes) e estão localizados em uma posição fixa no disco (área ou setor de boot).

  A contaminação por bloco de carga é simples: o vírus coloca seu código (ou parte dele) na posição do disco onde está o programa de *bootstrap* original e aloca uma outra região do disco para o restante do seu código (se necessário) e o programa de *bootstrap* original. Ao ser executado, o vírus aloca uma região de memória para si, desvia as rotinas de interrupção de tratamento de disco e carrega o programa de *bootstrap* original. A partir de então, a carga do sistema operacional ocorre normalmente.

  Dessa forma, ao assumir a posição de programa de *bootstrap* original, o vírus é carregado antes mesmo do sistema operacional, o que o impede de ter acesso às funções de mais alto nível implementadas pelo SO, acessando apenas as funções implementadas pelo BIOS da máquina.

  O método de propagação é simples: durante qualquer acesso a disco (leitura ou escrita), o SO executa a rotina de tratamento de disco do BIOS. Como o vírus desvia a rotina para si próprio, neste momento o vírus é ativado e tem a oportunidade de contaminar a área de boot do disco onde será realizada a operação de leitura/escrita. Após a contaminação, o vírus desvia o controle para a rotina de tratamento de disco original e a operação de leitura/escrita é executada normalmente. Este tipo de vírus não altera outros programas e se propaga através da área de boot dos discos "infectados".

  Vale salientar que a ordem pode não ser necessariamente "SO → ativa vírus → contamina → ativa rotina de disco original", mas também "SO → ativa vírus → ativa rotina de disco original → retorna ao vírus → contamina". No primeiro caso a contaminação é realizada antes do acesso ao disco, enquanto no outro caso a contaminação é realizada após o acesso ao disco.

- Envelopamento

  Ao contrário do vírus de *bootstrap*, o vírus de envelopamento atua sobre os programas executáveis, contaminando-os. O vírus altera estes programas, que passam a conter uma cópia do vírus. O início da execução do programa é desviado para a cópia do vírus, que após efetuar suas ações (instalação em memória, contaminação de outros programas, etc) retorna o controle ao programa original, que por fim é executado normalmente.

[1] *Imbedded Drive Electronics*.
[2] Em alguns micros, existe a opção do sistema operacional em ROM.
[3] *Basic Input/Output System*

Esta forma de contaminação é a mais complexa, uma vez que exige o conhecimento dos diversos tipos de programas executáveis, exigindo também o conhecimento do modo através do qual cada tipo é carregado na memória e executado. Vale ressaltar que em geral, tipos distintos de programas executáveis possuem formas bastante diferentes de carga e execução.

Uma forma de simplificar a implementação de um vírus deste tipo consiste na escolha de um um tipo de programa executável a ser contaminado que possua carga e execução simples, ou que seja mais comum.

A grande maioria dos vírus conhecidos adotam a contaminação por envelopamento. Em [Weber89] são descritos tipos de vírus inclusos nesta categoria, tais como: Vírus de Sobreposição, Vírus de Ligação, Vírus de Envelope (*Shell Virus*), Vírus Criptográfico, *Batch Virus* (também conhecidos como Vermes), etc.

4 A Semântica de Ações do Vírus Sexta-Feira 13

Um vírus sexta-feira 13 caracteriza-se por ser um vírus de envelopamento, contaminando portanto programas executáveis. No entanto, apenas micros do padrão IBM-PC com sistema operacional DOS[4] [IBM87] (ou compatível) são passíveis de contaminação pelo mesmo.

Na semântica formal do sexta-feira 13, com a finalidade de evitar *descrições* que na verdade pertencem a outros níveis, foi considerado que as funções, operações, e ligações de responsabilidade do sistema operacional, devem ter sua semântica formal descrita pela semântica de ações do próprio sistema operacional. Portanto, uma operação como:

```
"DOS.execute [ "Erase File" the String bound to DOS.Env_Programa_Corrente ]"
```

na semântica do vírus, indica que a função semântica execute e a ligação Env_Programa_Corrente estão definidas na semântica de ações do DOS. Dessa forma, o rótulo "NOME." será utilizado para indicar que a função (ou operação, ligação, etc) que segue "NOME." está definida na semântica de Ações de "NOME".

É importante observar que as informações mantidas pelo DOS estão associadas a ligações (ex.: o nome do arquivo que irá ou está sendo executado está na ligação Env_Programa_Corrente, o conteúdo do disco está na ligação Disk, etc). Assim, todas as funções semânticas do vírus são da forma rebind moreover ..., onde rebind reproduz as ligações fornecidas (evitando a perda das ligações fornecidas pelo DOS) e moreover faz com que as alterações realizadas pelo vírus nas ligações do DOS sejam passadas adiante.

Um programa é executado pelo DOS através da ativação de sua abstração, dada por:

```
DOS.execute [ "Load and Execute" file_name ] =
    | rebind
    moreover
    | | give (the Map bound to DOS.Disk) at the file_name
    | |     [should give an Abstraction]
    then
    | enact it
```

A descrição acima corresponde à operação do DOS que executa o arquivo "file_name" (tipo *String*). DOS.Disk pode ser visto como uma ligação cujo conteúdo é um mapeamento de nome de arquivos de dados/programas em suas abstrações.

[4] *Disk Operation System* - Sistema Operacional em Disco, utilizado por praticamente "todos" os micros compatíveis com o padrão IBM-PC

A forma escolhida para descrever a semântica de ações de uma das mutações de vírus sexta-feira 13 consiste em fornecer a semântica para cada ação executada pelo vírus ao se instalar e contaminar os outros programas. A descrição é baseada nas seguintes definições:

- Sintaxe Abstrata - contém o conjunto de operações e funções que serão executados e utilizados pelo vírus.
- Entidades Semânticas - contêm a definição dos domínios (*sorts*) e operações sobre estes domínios, definindo portanto uma álgebra. Como o vírus é executado no microprocessador 8086 [Intel79] e o sistema operacional DOS, é natural que todos os domínios e seus operadores estejam definidos na semântica do 8086 e do DOS.
- Funções Semânticas - contêm a definição das funções que fornecem o significado de cada elemento definido na sintaxe abstrata através do mapeamento de cada elemento em sua denotação (que é composta por ações e elementos das entidades semânticas).

A visão da estrutura geral da especificação é dada a seguir, com cada um dos itens detalhados nas sub-seções seguintes. A especificação está anotada o suficiente (**comment:** ...) com o objetivo de tornar a leitura do texto formal possível sem conhecimento muito detalhado da semântica de ações. Adicionalmente, o estilo verboso e modular da semântica de ações a torna intrinsecamente mais legível que muitos dos outros formalismos existentes.

(1) **VIRUS**

(1) **Abstract Syntax**

Virus-Actions ...;
Virus-Auxiliary-Actions ...;

(2) **Semantic Functions**

needs: Abstract Syntax, Semantic Entities;
The-Virus ...;
Virus-Actions ...;
Virus-Auxiliary-Actions ...;
Virus-Auxiliary-Interruption-Handlers ...;

(3) **Semantic Entities**

Standard ...;
CPU-Registers ...;
Interruption-Management ...;
Memory-Management ...;
File-Management ...;

A seguir, apresentamos apenas um resumo da especificação. A semântica completa da mutação descrita neste artigo pode ser encontrada em [Lin92].

4.1 Sintaxe Abstrata

Estas são as operações (ações) básicas realizadas pelo vírus. A sintaxe abstrata é dada seguindo a notação EBNF.

(1) **Virus/Abstract Syntax**

(1) **grammar:**

(2) **Virus-Actions**

Virus-Action =
"Instala Vírus"
| "Executa Programa sem esta Cópia do Vírus"
| "Contamina Programa a ser Executado"
| "Carrega e Executa Novamente o Programa"
| "Termina e Mantém o Código do Vírus Residente na Memória"
| "Elimina Arquivo a ser Executado"
| "Limpa Quadrado no Vídeo"
| "Laço para Consumir Tempo de CPU"
| "Desvia Interrupção 21h"
| "Desvia Interrupção 08h"
| "Desvia Interrupção 24h"
| "Executa Interrupção 21h Original"
| "Executa Interrupção 08h Original"
| "Restaura Interrupção 24h Original"
□

(3) **Virus-Auxiliary-Actions**

Virus-Auxiliary-Action =
"Vírus Instalado ?"
| "É arquivo COM ?"
| "Arquivo é o COMMAND.COM?"
□

4.2 Entidades Semânticas

Nesta seção são definidos os elementos básicos da especificação: domínios (*sorts*) e seus operadores.

(1) **Virus/Semantic Entities/Standard**

comment: Contido na semantica de ações do 8086.
Define os tipos de dados e os elementos passíveis de serem armazenados (tipo *storable*) em células de memória;

...

(2) **Virus/Semantic Entities/CPU-Registers**

comment: Contido na semantica de ações do 8086.
Define os registradores do 8086, fornecendo ainda formas de acesso ao byte mais significativo e menos significativo dos registradores AX, BX, CX e DX;

notation:
BP, SP, IP, DI, SI, CS, DS, SS, ES,
AL, AH, BL, BH, CL, CH, DL, DH, FLAGS.

...

(3) **Virus/Semantic Entities/Interruption-Management**

comment: Definição contida na semantica de ações do 8086.
Gerenciamento contido na semantica de ações do DOS.
Define o tipo interrupção e as formas de acesso às interrupções do 8086;

notation:
interruption, interruption _, abstraction of _.

...

(4) **Virus/Semantic Entities/Memory-Mangement**

comment: Contido na semântica de ações do DOS.
Define o tipo Memória e fornece meios de acesso (alocar, desalocar, carga de programas para execução, ...);

...

(5) **Virus/Semantic Entities/File-Management**

comment: Contido na semântica de ações do DOS.
Define o tipo Disco e os respectivos métodos de acesso (leitura, escrita, formatação, ...);

...

4.3 Funções Semânticas

Esta seção contém a definição das funções semânticas que fornecem o significado de cada ação do vírus dada na sintaxe abstrata (seção 4.1) em termos de elementos das entidades semânticas (seção 4.2)

(1) **Virus/Semantic Functions/The-Virus**

comment: É a definição mais genérica do vírus, onde ele testa se está instalado na memória. Caso esteja, executa o programa original sem esta cópia do vírus; caso contrário o sistema é contaminado;

notation:
run;

```
run :: action [binding].
run =
    | rebind
    moreover
    | | give abstraction
    | | | | evaluate [| "Vírus Instalado ?" |] [must give a Truth-value]
    | | | then
    | | | | | | check (it is true)
    | | | | then
    | | | | | | store DDh in 8086 AH
    | | | | | and then
    | | | | | | 8086 execute [| "INT" 21h |]
    | | | | or
    | | | | | check (it is false) then execute [| "Instala Virus" |]
    | | then
    | | enact it
```

comment: Observe em "Virus/Semantic Functions/Virus-Auxiliary-Interruption-Handlers" que na abstração da nova interrupção 21h, a função DDh é utilizada pelo vírus para executar o programa original antes desta contaminação ;

(2) **Virus/Semantic Functions/Virus-Actions**

comment: Define o significado de cada ação realizada pelo vírus;

notation:
execute [| _ |];

```
execute :: virus-actions -> action [binding]
```

```
execute [ "Instala Vírus" ] =
    | rebind
    moreover
    | | | bind 31813 to dContador_Int_08h
    | | and then
    | | | | execute [ "Desvia Interrupção 21h" ]
    | | | and
    | | | | execute [ "Desvia Interrupção 08h" ]
    | | and then
    | | | execute [ "Carrega e Executa Novamente o Programa" ]
    | | and then
    | | | execute [ "Termina e Mantém o Código do Vírus Residente na Memória" ]
```

comment: O valor numérico ligado ao identificador dContador_Int_08h será utilizado pelo vírus como a quantidade de interrupções geradas pelo relógio (*ticks*) a serem contadas antes que o desempenho do equipamento seja diminuído. É importante observar que após o desvio das interrupções 21h (serviços do DOS) e 08h (relógio), o programa é executado novamente (portanto, na instalação do vírus na memória, o programa é carregado para execução duas vezes). Após a execução do programa, o vírus finalmente executa a ação para manter o seu código residente;

```
execute [ "Executa Programa sem esta Cópia do Vírus" ] =
    | rebind
    moreover
    | enact (the Abstration bound to dPrograma_Original)
```

comment: Para executar o programa original o vírus habilita (*enact*) a abstração que corresponde à semântica do programa antes desta contaminação. A abstração ligada ao identificador dPrograma_Original é obtida antes que esta cópia do vírus seja acrescentada ao arquivo (cf. execute ["Contamina Programa a ser Executado"]);

```
execute [ "Contamina Programa a ser Executado" ] =
    | rebind
    moreover
    | | | | give DOS evaluate [ "Read File" the String bound to DOS Env_Programa_a_
    | | | |     ser_Executado ] [must give an Abstration]
    | | | then
    | | | | bind it to dPrograma_Original
    | | and
    | | | bind (the String bound to DOS Env_Programa_a_ser_Executado) to dPrograma_
    | | |     Temp
    | hence
    | | | | execute [ "Desvia Interrupção 24h" ]
    | | | and then
    | | | | DOS Execute [ "Write File" (the String bound to dPrograma_Temp)
    | | | |     (closure abstration THE-VIRUS) ]
    | | and then
    | | | execute [ "Restaura Interrupção 24h Original" ]
```

comment: Em uma ação "A1 hence A2" as ligações produzidas por A1 são passadas apenas para A2 (neste caso, dPrograma_Original e dPrograma_Temp). As ligações produzidas pela ação composta são aquelas produzidas por A2 (aqui, nenhuma ligação é produzida pela ação composta pelo hence). O closure é essencial pois é quem irá garantir que quando a abstração do vírus for habilitada o identificador dPrograma_Original estará ligado à abstração do programa antes da contaminação. Observe ainda que não é realizado nenhum teste para saber se o arquivo já foi contaminado anteriormente, portanto o vírus contamina o programa toda vez que é executado. Outra observação importante é que o programa é contaminado antes da execução e não há preocupação em restaurar a hora/dia de última alteração, portanto no diretório o arquivo aparece com a data e hora da contaminação mais recente e não há uma preocupação em modificar o atributo do arquivo. Desta forma, arquivos protegidos contra escrita e/ou ocultos não poderão ser contaminados;

```
execute [ "Carrega e Executa Novamente o Programa" ] =
    | rebind
    moreover
    | DOS.execute [ "Load and Execute" (the String bound to
    |     DOS.Env_Programa_Corrente) ]
execute [ "Termina e Mantém o Código do Vírus Residente na Memória" ] =
    | rebind
    moreover
    | DOS.execute [ "Terminate and Stay Resident" ]
execute [ "Elimina Arquivo a ser Executado" ] =
    | rebind
    moreover
    | DOS.execute [ "Erase File" (the String bound to
    |     DOS.Env_Programa_Corrente) ]
execute [ "Limpa Quadrado no Vídeo" ] =
    | rebind
    moreover
    | BIOS.execute [ "Scrool Screen Up" 06 06 17 17 ]
```

comment: As quatro ações acima não trazem nada de novo visto que correspondem às funções implementadas pelo sistema operacional e pelo BIOS;

```
execute [ "Laço para Consumir Tempo de CPU" ] =
    | rebind
    moreover
    | | | bind 16385 to dContador
    | | then
    | | | unfolding
    | | | | | check ((the Number bound to dContador is 0) is false)
    | | | | then
    | | | | | bind (predecessor(the Byte bound to dContador_Int_08h) to
    | | | | |     dContador
    | | | | then
    | | | | | unfold
    | then
    | | complete
```

comment: Como sugerido pelo nome, a ação acima é utilizada pelo vírus para diminuir o desempenho do equipamento através de um laço inútil de 16.385 passos;

```
execute [ "Desvia Interrupção 21h" ] =
    | rebind
    moreover
    | | | DOS.evaluate [ "Get Int Vector" 21h ] [must give an Interruption]
    | | then
    | | | bind it to dInt_21h_Original
    | and then
    | | DOS.execute [ "Set Int Vector" 21h (the Interruption bound to
    | |     dNova_Int_21h) ]
```

```
execute [ "Desvia Interrupção 08h" ] =
    | rebind
    moreover
    | | | DOS.evaluate [ "Get Int Vector" 08h ] [must give an Interruption]
    | | then
    | | | bind it to dInt_08h_Original
    | and then
    | | DOS execute [ "Set Int Vector" 08h (the Interruption bound to
    | |     dNova_Int_08h) ]
execute [ "Desvia Interrupção 24h" ] =
    | rebind
    moreover
    | | | DOS.evaluate [ "Get Int Vector" 24h ] [must give an Interruption]
    | | then
    | | | bind it to dInt_24h_Original
    | and then
    | | DOS execute [ "Set Int Vector" 24h (the Interruption bound to
    | |     dNova_Int_24h) ]
```

comment: As ações para desvio de interrupção obtem a interrupção atual, ligam-na a um identificador e modificam o vetor de interrupção para que a rotina de interrupção a ser executada seja a implementada pelo vírus;

```
execute [ "Executa Interrupção 21h Original" ] =
    | rebind
    moreover
    | enact (the Interruption bound to dInt_21h_Original)
execute [ "Executa Interrupção 08h Original" ] =
    | rebind
    moreover
    | enact (the Interruption bound to dInt_08h_Original)
```

comment: Para executar a interrupção original basta habilitar a interrupção ligada ao identificador apropriado;

```
execute [ "Restaura Interrupção 24h Original" ] =
    | rebind
    moreover
    | DOS execute [ "Set Int Vector" 24h (the Interruption bound to dInt 24h Original) ]
```

comment: Para restaurar a interrupção original basta que a interrupção ligada ao identificador apropriado seja recolocada no vetor de interrupção;

(3) **Virus/Semantic Functions/Virus-Auxiliary-Actions**

comment: Contém a definição das funções auxiliares ...

notation:
evaluate [_].

evaluate : virus-auxiliary-actions → action [giving a result].
comment: result = Truth Value | Natural | String

```
evaluate [ "Vírus Instalado ?" ] =
    | store E0h in 8086 AH then 8086 execute [ "INT" 21h ]
    and then
    | | currently stored in 8086 AH is 03h
    | then
    | | give it [must give a Truth Value]
```

comment: Esta função é utilizada pelo vírus para verificar a sua presença na memória do equipamento;

```
evaluate [ "É arquivo COM ?" ] =
    | rebind
    moreover
    | | is (".COM") in (the String bound to DOS.Env.Programa Corrente)
    | then
    | | give it [must give a Truth-Value]

evaluate [ "Arquivo é o COMMAND.COM?" ] =
    | is ("COMMAND.COM") in (the String bound to DOS.Env.Programa.Corrente)
    then
    | give it [must give a Truth-Value]
```

comment: Nas duas funções anteriores, a operação "is S11 in S12" é uma operação do domínio *String* definida em "Virus/Semantic Entities/Standard". A operação retorna como resultado um elemento do domínio *Truth-Value* condicionalmente ao *String* ST1 estar ou não contido no *String* ST2;

(4) Virus/Semantic Functions/Virus-Auxiliary-Interruption-Handlers

```
abstracao_nova_int_21h =
    bind dNova_Int_21h to interruption
        | | | rebind
        | | moreover
        | | | | check (disjuntion (check currently stored in 8086 AH is 4Bh,
        | | | |     disjuntion (check currently stored in 8086 AH is E0h,
        | | | |     currently stored in check 8086 AH is DDh)) is false)
        | | | then
        | | | | complete
        | | or
        | | | check (currently stored in 8086 AH is E0h) then store 03h in 8086 AH
        | | or
        | | | | check (currently stored in 8086 AH is DDh)
        | | | then
        | | | | execute [ "Executa Programa sem esta Cópia do Vírus" ]
        | | or
        | | | | check (currently stored in 8086 AH is 4Bh)
        | | | then
        | | | | | give conjuction (DOS Env Dia Mes is 13, conjuction (
        | | | | |     DOS Env Dia Semana is 'SEXTA', not (DOS Env Ano is 1987)))
        | | | | then
        | | | | | | check (it is true)
        | | | | | then
        | | | | | | execute [ "Elimina Arquivo a ser Executado" ]
        | | | | or
        | | | | | | check (it is false)
        | | | | | then
        | | | | | | execute [ "Contamina Programa a ser Executado" ]
        | | and then
        | | execute [ "Executa Interrupção 21h Original" ]
```

comment: O vírus implementa apenas as funções 4Bh, E0h e DDh da interrupção 21h. A função 4Bh é importante pois é a responsável pela carga e execução dos programas. Assim, o vírus tem total controle sobre os programas que serão executados. Observe que em uma sexta-feira 13, o arquivo que deveria ser executado é eliminado do disco antes da execução (respeitando a condição que o ano seja diferente de 1987), nos outros dias o arquivo é apenas contaminado. A função E0h é utilizada pelo vírus para informar as outras cópias presentes nos arquivos que o vírus já está instalado no equipamento.

A função DDh é utilizada para executar o programa sem esta cópia do vírus (pela ação "Contamina Programa a ser Executado" é fácil ver que um programa pode ser contaminado diversas vezes):

```
abstracao.nova_int_08h =
    bind dNova_Int_08h to interruption
        | | rebind
        moreover
        | | give (the Byte bound to dContador_Int_08h is greater than 02)
        | |     [must give a Truth-Value]
        then
        | | | check (it is true)
        | | then
        | | | bind predecessor(the Byte bound to dContador_Int_08h) to
        | | |     dContador_Int_08h
        | | or
        | | | check (it is false)
        | | then
        | | | give (the Byte bound to dContador_Int_08h is 02)
        | | |     [must give a Truth-Value]
        | | then
        | | | | check (it is true)
        | | | then
        | | | | execute [ "Limpa Quadrado no Vídeo" ]
        | | | | and
        | | | | bind 01 to dContador_Int_08h
        | | | or
        | | | | check (it is false)
        | | | then
        | | | | execute [ "Laço para Consumir Tempo de CPU" ]
        and then
        | execute [ "Executa Interrupção 08h Original" ]
```

comment: Esta abstração da interrupção 08h é habilitada pelo *hardware* do equipamento a cada pulso do relógio (*tick*) a uma razão de aproximadamente 18,2 vezes/segundo. O identificador dContador_Int_08h está ligado ao número de *ticks* que devem ocorrer sem que o vírus tome alguma ação. Se o seu valor for 2, o vírus limpa uma região do monitor e quando chegar a 1, a cada *tick* é executado uma ação ("Laço para Consumir Tempo de CPU") cujo objetivo é diminuir o desempenho do equipamento;

```
abstracao.nova_int_24h =
    bind dNova_Int_24h to interruption
        | rebind
        moreover
        | store 00 in 8086 AL
```

comment: Após a terceira tentativa de acesso a um dispositivo de entrada/saída sem sucesso (por exemplo, disco), o Sistema Operacional passa o controle para a interrupção 24h. Esta, ao retornar, deve informar ao DOS através do registrador AL, qual ação deve ser tomada: ignorar o erro (AL=0), tentar novamente a operação (AL=1), terminar a execução do programa corrente através da interrupção 23h (*exit*, AL=2) ou sinalizar uma falha no sistema (AL=3). Como o vírus não deseja que o usuário perceba que houve uma tentativa mal sucedida de acesso ao disco (por exemplo, disco protegido contra escrita), a implementação da interrupção 24h do vírus informa ao sistema operacional que o erro deve ser ignorado;

5 Conclusões e Direções Futuras

O estilo de descrição da semântica de ações tornou mais fácil e clara a compreensão da especificação, simplificada pelo alto nível de modularidade conseguido. Através da semântica formal apresentada, tornou-se possível detectar algumas "falhas" nesta versão do vírus sexta-feira 13, tais como:

- O vírus não é capaz de contaminar arquivos protegidos contra escrita e/ou ocultos;
- Ao contaminar um arquivo, o vírus não restaura a data e hora da última alteração, tornando possível a sua detecção, uma vez que a data e hora da contaminação são registradas automaticamente pelo DOS e substituem as informações originais;
- O vírus não verifica a contaminação prévia de um arquivo. Sendo assim, como os arquivos ".COM" possuem um limite de 64Kb para tamanho físico, após sucessivas contaminações os mesmos não mais poderão ser executados, o que conduz a uma rápida detecção. Com a múltipla contaminação, a carga dos programas torna-se excessivamente lenta e a redução do espaço disponível em disco também ocorre de forma rápida, sinalizando a presença do vírus.

Este trabalho, além de permitir que o combate ao vírus seja mais efetivo, por proporcionar o conhecimento detalhado de suas ações, direcionou outras linhas de pesquisa interessantes. Sua continuidade visa o término do desenvolvimento da semântica de ações do sistema operacional DOS e do microprocessador 8086. O ambiente obtido através da definição semântica dos diversos níveis de abstração presentes nos micros IBM-PC com sistema DOS, permitirá a descrição e análise semântica de classes de programas que podem ser executados utilizando tais recursos.

Semântica de ações se provou uma ferramenta eficiente para descrição de sistemas de classe dos vírus, sendo o entendimento das descrições formais acessível mesmo para os não iniciados. Como este foi um dos objetivos do projeto de Mosses, pode-se dizer que neste ponto ele é bem sucedido.

A disponibilidade eventual de um simulador de descrições como a aqui apresentada tornaria possível a execução parcial da especificação, possivelmente aumentando a compreensão e facilitando a escrita semântica.

Referências

[Bergstra89] **J. A. Bergstra, J. Heering, P. Klint**: "ALGEBRAIC SPECIFICATION", *Addison Wesley*, 1989.

[Burstall77] **R. Burstall, J. Goguen**: "PUTTING THEORIES TOGETHER TO MAKE SPECIFICATIONS", *Proc. Fifth Int'l Joint Conf. Artificial Intelligence*, Cambridge, Mass. 1977.

[Fainberg89] **T. Fainberg**: "THE NIGHT THE NETWORK FAILED", *New Scientist, 38-42*, March 4, 1989.

[Goguen78] **J.A. Goguen, J.W. Thatcher, E.G. Wagner**: "AN INITIAL ALGEBRA APPROACH TO THE SPECIFICATION, CORRECTNESS AND IMPLEMENTATION OF ABSTRACT DATA TYPES", *Current Trends in Programming Methodology, vol. IV, 80 -149*, Prentice-Hall, 1978.

[Goguen84] **J.A. Goguen**: "PARAMETERIZED PROGRAMMING", *IEEE Transactions on Software Engineering, vol. 5, no. SE-10*, September 1984.

[Greenberg89] **R.M. Greenberg**: "KNOW THY VIRAL ENEMY", *Byte, 275-284*, June 1989.

[Guessarian81] **I. Guessarian**: "ALGEBRAIC SEMANTICS", *Lecture Notes in Computer Science vol. 99*, Springer Verlag, 1982.

[Gunter89] **C.A. Gunter, P.D. Mosses, D.S. Scott**: "SEMANTIC DOMAINS AND DENOTATIONAL SEMANTICS", *MS-CIS-89-16 Logic and Computation 04*, University of Pennsylvania, February 1989.

[Hoare71] **C.A.R. Hoare**: "PROCEDURES AND PARAMETERS, AN AXIOMATIC APPROACH", *Lecture Notes in Mathematics, no. 188, 102-116*, Springer Verlag, 1971.

[IBM87] **IBM - International Business Machines Corporation**: "DISK OPERATING SYSTEM VERSION 3.30", *Technical Reference/Upgrade*, 1987.

[Intel79] **Intel Corporation**: "THE 8086 FAMILY - USER'S MANUAL", 1979.

[Lin92] **Lin T.M., S.R.L. Meira**: "A SEMÂNTICA DE VÍRUS DE COMPUTADOR", *Relatório Técnico, DI-UFPE*, 1992.

[McAfee92] **McAfee Associates**: "VIRUS CHARACTERISTICS LIST V93", *Software Documentation*, 1992.

[Minasi91] **M. Minasi**: "COMPUTER VIRUSES FROM A TO Z", *COMPUTE, 45-49*, COMPUTE Publications International Ltd, October, 1991.

[Mismetti90] **F. Mismetti, C. Palludetti, N.S. Mismetti**: " VÍRUS - GUIA DE REFERÊNCIA TÉCNICA", *McGraw-Hill*, 1990.

[Mosses88] **P.D. Mosses**: "THE MODULARITY OF ACTION SEMANTICS", *Internal Report - DAIMI IR - 75* Aarhus University, Dinamarca, 1988.

[Mosses89] **P.D. Mosses**: "A PRACTICAL INTRODUCTION TO DENOTATIONAL SEMANTICS", *State of the Art Seminar on Formal Description of Programming Concepts*, Rio de Janeiro, Brasil, 1989.

[Mosses89a] **P.D. Mosses**: "UNIFIED ALGEBRAS AND ACTION SEMANTICS", *Proc. Symposium on Theoretical Aspects of Computer Science*, 1989.

[Mosses89b] **P.D. Mosses**: "ACTION SEMANTICS", *Draft, Version 7* Aarhus University, Dinamarca, 1989.

[Musicante90] **M.A. Musicante, S.R.L. Meira, R.D.Lins**: "A SEMÂNTICA DE AÇÕES DE GM-C", *Anais X Simpósio da SBC*, 1990.

[Plotkin81] **G.D. Plotkin** : "A STRUCTURAL APPROACH TO OPERATIONAL SEMANTICS", *Internal Report - DAIMI FN-19*, Aarhus University, Denmark, Sep 1981.

[Rubenking89] **N.J. Rubenking**: "INFECTION PROTECTION", *PC Magazine, 194-227*, April 25, 1989.

[Ruber91] **P. Ruber**: "VIRUS KILLERS", *Computer Monthly, vol. 1, no. 9, 99-100*, Volcan Publications Inc, 1991.

[Schmidt86] **D.A. Schmidt**: "DENOTATIONAL SEMANTICS: A METHODOLOGY FOR LANGUAGE DEVELOPMENT", Allyn and Bacon, 1986.

[Scott82] **D.S. Scott**: "DOMAINS FOR DENOTATIONAL SEMANTICS", *Lecture Notes in Computer Science vol. 140, 577-613*, Springer, 1982.

[Seymour88] **J. Seymour, J. Matzkin**: "CONFRONTING THE GROWING THREAT OF HARMFUL COMPUTER SOFTWARE VIRUSES", *PC Magazine, 33-36*, June 28, 1988.

[Stoy79] **J. E. Stoy**: "FOUNDATIONS OF DENOTATIONAL SEMANTICS", *Abstract Software Specifications - Kobenhavn School Proc.*, D. Bjorner, 1979.

[Watt90] **D.A. Watt**: "PROGRAMMING LANGUAGE SYNTAX AND SEMANTICS", *Draft* University of Glasgow, Dec 11, 1990.

[Weber89] **R.F. Weber**: "VÍRUS DE COMPUTADOR", *Revista de Informática Teórica e Aplicada, vol. 1, no. 1*, 1989.