On semantics of self-modifying codes

Hubert Godfroy

CARTE team INRIA Nancy

May 25, 2013

Team presentation

- Laboratoire lorrain de recherche en informatique et ses applications (LORIA) à Nancy
- CARTE team: Calculablity & complexity, virology

Self-modification

Self-modification is the ability of program to execute data he wrote himself

Self-modification

Self-modification is the ability of program to execute data he wrote himself

Assembly

```
1:  sub 2 42
2:  𝔻(𝔼(mov eax 18) + 42)
```

Self-modification

Self-modification is the ability of program to execute data he wrote himself

Assembly

```
1:   sub 2 42
2:   𝔻(𝔼(mov eax 18) + 42)
```

ML

```
let f = ref λ x.x in (!f) 2
```

Self-modification

Self-modification is the ability of program to execute data he wrote himself

Assembly

```
1:   sub 2 42
2:   𝔻(𝔼(mov eax 18) + 42)
```

ML

```
let f = ref λ x.x in (!f) 2
```

- Does it exist framworks to study self-modifying codes?

Level of execution

For each address is associated its execution level:

> *the execution level of an instruction is the execution level of the instruction which wrote it, plus 1.*

Level of execution

For each address is associated its execution level:

> *the execution level of an instruction is the execution level of the instruction which wrote it, plus 1.*

```
1:  sub 2 42
2:  sub 3 42
3:  𝔻(𝔼(mov eax 18) + 42)
4:  𝔻(𝔼(mov 5 𝔼(jump 666)) + 42)
5:  add eax ebx ecx
```

Level of execution

For each address is associated its execution level:

the execution level of an instruction is the execution level of the instruction which wrote it, plus 1.

```
1:  sub 2 42                              1
2:  sub 3 42                              1
3:  𝔻(𝔼(mov eax 18) + 42)
4:  𝔻(𝔼(mov 5 𝔼(jump 666)) + 42)
5:  add eax ebx ecx
```

Level of execution

For each address is associated its execution level:

> *the execution level of an instruction is the execution level of the instruction which wrote it, plus 1.*

1:	`sub 2 42`	1
2:	`sub 3 42`	1
3:	$\mathbb{D}$(`mov eax 18`) `+ 42`) → $\mathbb{D}(\mathbb{E}(\texttt{mov eax 18}) + \texttt{42})$	2
4:	$\mathbb{D}(\mathbb{E}(\texttt{mov 5 } \mathbb{E}(\texttt{jump 666})) + \texttt{42})$	2
5:	`add eax ebx ecx`	

Level of execution

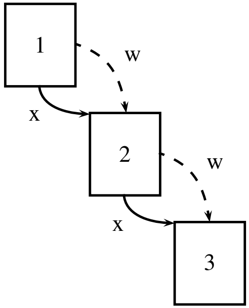
For each address is associated its execution level:

> *the execution level of an instruction is the execution level of the instruction which wrote it, plus 1.*

```
1:   sub 2 42                          1
2:   sub 3 42                          1
3:   𝔻(𝔼(mov eax 18) + 42)             2
4:   𝔻(𝔼(mov 5 𝔼(jump 666)) + 42)      2
5:   add eax ebx ecx                   3
```

Waves of self-modification

A wave is the set of addresses with the same execution level



Remaining questions

- Does it exist a semantic explaining waves?
- Is it possible to build a wave classification to specify compilers to self-modifying codes?
- Is there any existing framework which could catch self-modification semantics?

Idea for further developement

- Waves switches $\sim$ CPS
- Abstract machines (Kripke or Curien-Herbelin-Wadler machine)
- Link with LK?