Abstract Self Modifying Machines

Hubert Godfroy

joint work with Jean-Yves Marion
Loria Nancy

October 14, 2014

Plan

Plan

Introduction

Framework

Applications

Program? Data?

- A program is something which can be executed.
- A data is something which can be read and write

Program? Data?

- A program is something which can be executed.
- A data is something which can be read and write

- In most langages, programs are distinct from data.
- Example : C, JAVA, OCaml

Program? Data?

- A program is something which can be executed.
- A data is something which can be read and write

- In most langages, programs are distinct from data.
- Example : C, JAVA, OCaml

- There is exceptions…
- Programs with `exec` function have self-modifying behaviors
- Example : Python

Program? Data?

- A program is something which can be executed.
- A data is something which can be read and write

- In most langages, programs are distinct from data.
- Example : C, JAVA, OCaml

- There is exceptions…
- Programs with `exec` function have self-modifying behaviors
- Example : Python

Low level case: nothing is forbidden!

- Programs and data are totally indistinguishable
- They belong to the same space (memory)

Example

```
 1  move 10 2
 2  jz $\langle 10 \rangle$ 6
 3  move$(10 - \langle 10 \rangle)$ $((\langle 10 - \langle 10 \rangle \rangle) - 42)$
 4  move 10 $(\langle 10 \rangle - 1)$
 5  jump 2
 6  jump 8
 7  stop
 8  $(\mathbb{E}($print hello world$) + 42)$
 9  $(\mathbb{E}($jump 7$) + 42)$
10  jump 1
```

Example

```
1  $\texttt{move}\ 10\ 2$
2  $\texttt{jz}\ \langle 10\rangle\ \ 6$
3  $\texttt{move}(10 - \langle 10\rangle)\ (\langle 10 - \langle 10\rangle\rangle) - 42$
4  $\texttt{move}\ 10\ (\langle 10\rangle - 1)$
5  $\texttt{jump}\ 2$
6  $\texttt{jump}\ 8$
7  $\texttt{stop}$
8  $(\mathbb{E}(\texttt{print}\ \textbf{hello world}) + 42)$
9  $(\mathbb{E}(\texttt{jump}\ 7) + 42)$
10 $\texttt{jump}\ 1$
```

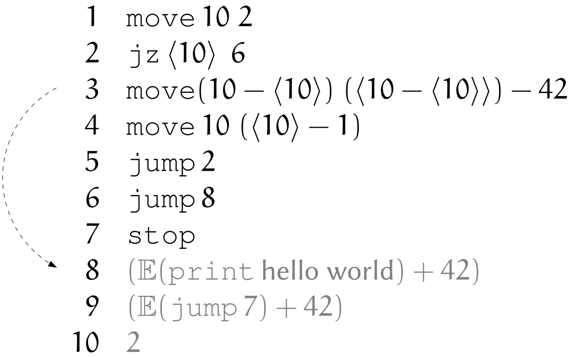
Example

1 $\texttt{move}\ 10\ 2$
2 $\texttt{jz}\ \langle 10\rangle\ \ 6$
3 $\texttt{move}(10-\langle 10\rangle)\ (\langle 10-\langle 10\rangle\rangle)-42$
4 $\texttt{move}\ 10\ (\langle 10\rangle-1)$
5 $\texttt{jump}\ 2$
6 $\texttt{jump}\ 8$
7 $\texttt{stop}$
8 $(\mathbb{E}(\texttt{print}\ \textsf{hello world})+42)$
9 $(\mathbb{E}(\texttt{jump}\ 7)+42)$
10 $\texttt{jump}\ 1$

Example

```
 1  move 10 2
 2  jz ⟨10⟩ 6
 3  move(10 − ⟨10⟩) (⟨10 − ⟨10⟩⟩) − 42
 4  move 10 (⟨10⟩ − 1))
 5  jump 2
 6  jump 8
 7  stop
 8  (𝔼(print hello world) + 42)
 9  (𝔼(jump 7) + 42)
10  2
```

Example



1 $\texttt{move}\ 10\ 2$

2 $\texttt{jz}\ \langle 10\rangle\ \ 6$

3 $\texttt{move}(10 - \langle 10\rangle)\ (\langle 10 - \langle 10\rangle\rangle) - 42$

4 $\texttt{move}\ 10\ (\langle 10\rangle - 1)$

5 $\texttt{jump}\ 2$

6 $\texttt{jump}\ 8$

7 $\texttt{stop}$

8 $(\mathbb{E}(\texttt{print}\ \textbf{hello world}) + 42)$

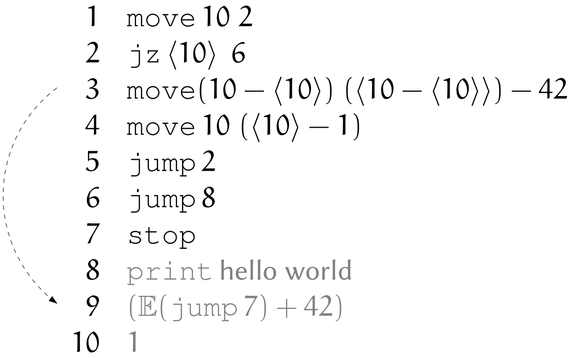
9 $(\mathbb{E}(\texttt{jump}\ 7) + 42)$

10 2

Example

1 $\texttt{move}\ 10\ 2$
2 $\texttt{jz}\ \langle 10\rangle\ \ 6$
3 $\texttt{move}(10-\langle 10\rangle)\ (\langle 10-\langle 10\rangle\rangle)-42$
4 $\texttt{move}\ 10\ (\langle 10\rangle-1)$
5 $\texttt{jump}\ 2$
6 $\texttt{jump}\ 8$
7 $\texttt{stop}$
8 $\texttt{print}$ **hello world**
9 $(\mathbb{E}(\texttt{jump}\ 7)+42)$
10 2

Example



```
1   move 10 2
2   jz ⟨10⟩ 6
3   move(10 − ⟨10⟩) (⟨10 − ⟨10⟩⟩) − 42
4   move 10 (⟨10⟩ − 1)
5   jump 2
6   jump 8
7   stop
8   print hello world
9   ($\mathbb{E}$(jump 7) + 42)
10  1
```

Example

```
 1  move 10 2
 2  jz ⟨10⟩ 6
 3  move(10 − ⟨10⟩) (⟨10 − ⟨10⟩⟩) − 42
 4  move 10 (⟨10⟩ − 1)
 5  jump 2
 6  jump 8
 7  stop
 8  print hello world
 9  jump 7
10  0
```

Example

```
1   move 10 2
2   jz $\langle 10 \rangle$ 6
3   move $(10 - \langle 10 \rangle)$ $(\langle 10 - \langle 10 \rangle \rangle) - 42$
4   move 10 $(\langle 10 \rangle - 1)$
5   jump 2
6   jump 8
7   stop
8   print hello world
9   jump 7
10  0
```

Example

```
 1  move 10 2
 2  jz ⟨10⟩ 6
 3  move (10 − ⟨10⟩) (⟨10 − ⟨10⟩⟩) − 42
 4  move 10 (⟨10⟩ − 1)
 5  jump 2
 6  jump 8
 7  stop
 8  print hello world
 9  jump 7
10  0
```

Example

```
1   move 10 2
2   jz ⟨10⟩ 6
3   move(10 − ⟨10⟩) (⟨10 − ⟨10⟩⟩) − 42
4   move 10 (⟨10⟩ − 1)
5   jump 2
6   jump 8
7   stop
8   print hello world
9   jump 7
10  0
```

Example

```
1   move $10$ $2$
2   jz $\langle 10\rangle$ $6$
3   move$(10 - \langle 10\rangle)$ $(\langle 10 - \langle 10\rangle\rangle) - 42$
4   move $10$ $(\langle 10\rangle - 1)$
5   jump $2$
6   jump $8$
7   stop
8   print hello world
9   jump $7$
10  $0$
```

General problematics

Compilation & certification

- From non SM programs to SM programs (obfuscation/optimisation)
- Certify compilation

General problematics

Compilation & certification

- From non SM programs to SM programs (obfuscation/optimisation)
- Certify compilation

Recover high-level semantics from low-level SM semantics

- Recover non SM program from SM program…
- …wrt existing models of self-modification (wave semantics)

General problematics

Compilation & certification

- From non SM programs to SM programs (obfuscation/optimisation)
- Certify compilation

Recover high-level semantics from low-level SM semantics

- Recover non SM program from SM program…
- …wrt existing models of self-modification (wave semantics)

Program abstraction

- Find abstract model specifically taking about self-modification.

Plan

Introduction

Framework

Applications

Current frameworks

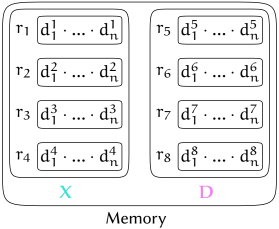
- Turing machine
- RAM (Cook & Reckhow, 1973)
- Cellular automaton (Neumann, 1966)
- Blob (Jones, 2010)
- RASP (Elgot & Robinson, 1964)
- SRM (Marion, 2012)

Language ASM$_2$

Language over data in $\mathcal{D}$, addresses in $\mathcal{A}$ and registers in $\mathcal{R}$:
$\forall r \in \mathcal{R}, \langle r \rangle : \mathcal{A} \to \mathcal{D}$

Abstract machine

- Register pointer: $\mathtt{RP} \in \mathcal{R}$
- Instruction pointer: $\mathtt{IP} \in \mathcal{A}$
- Executable zone: $\mathbf{X} \in \wp(\mathcal{R})$

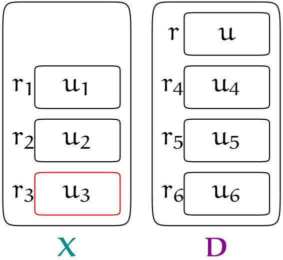


Instruction

The set of data $\mathcal{D}$ contains codes of the following instructions:

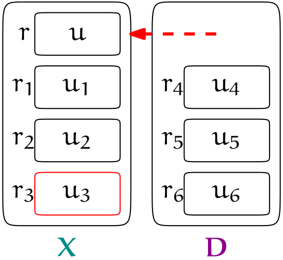
Instruction	Meaning
`move` r, d	Write the data d at the end of $\mathbf{D}[r]$
`input` r	Write the top of the input at the end of $\mathbf{D}[r]$
`pop` r	Pop the data on the top of $\mathbf{D}[r]$
`jump` a	Go to the instruction at address a
`case` r	Conditional jump depending on $\mathbf{D}[r]$
`exec` r	Control transfer to register $\mathtt{RP} = r$ and $\mathtt{IP} = 0$
`activate` r	Activate $\mathbf{D}[r]$
`inactivate` r	Inactivate $\mathbf{X}[r]$

Instruction `activate`



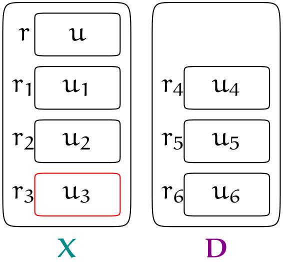
$$\begin{aligned} \texttt{RP} &= r_3 \\ \langle \texttt{RP} \rangle\, \texttt{IP} &= \texttt{activate}\; r \end{aligned}$$

Instruction `activate`



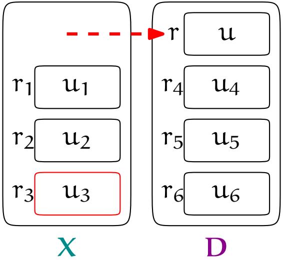
$$\begin{aligned} \texttt{RP} &= r_3 \\ \langle \texttt{RP} \rangle\, \texttt{IP} &= \texttt{activate}\, r \end{aligned}$$

Instruction `inactivate`



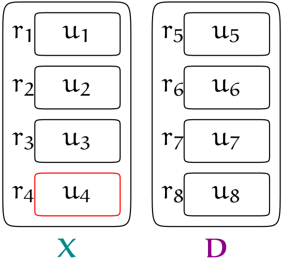
$$
\begin{aligned}
\mathtt{RP} &= r_3 \\
\langle\mathtt{RP}\rangle\,\mathtt{IP} &= \mathtt{inactivate}\,r
\end{aligned}
$$

Instruction `inactivate`



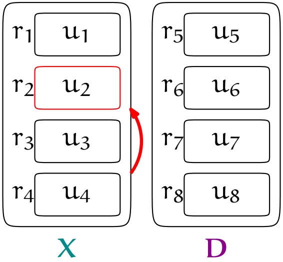
$$\begin{aligned} \mathtt{RP} &= r_3 \\ \langle \mathtt{RP} \rangle\, \mathtt{IP} &= \mathtt{inactivate}\, r \end{aligned}$$

Instruction `exec`



$$\begin{aligned} \mathtt{RP} &= r_3 \\ \langle \mathtt{RP} \rangle\, \mathtt{IP} &= \mathtt{exec}\, r \end{aligned}$$

Instruction exec



$$\begin{aligned}\mathtt{RP} &= r_3\\ \langle\mathtt{RP}\rangle\,\mathtt{IP} &= \mathtt{exec}\,r\end{aligned}$$

Example: decrypting code

```
 1  move 10 2
 2  jz $\langle 10 \rangle$ 6
 3  move$(10 - \langle 10 \rangle)$ $(\langle 10 - \langle 10 \rangle \rangle) - 42$
 4  move 10 $(\langle 10 \rangle - 1)$
 5  jump 2
 6  jump 8
 7  stop
 8  $(\mathbb{E}($print hello world$) + 42)$
 9  $(\mathbb{E}($jump 7$) + 42)$
10  jump 1
```

Example: decrypting code

```
1   move 10 2
2   jz ⟨10⟩ 6
3   move (10 − ⟨10⟩) (⟨10 − ⟨10⟩⟩) − 42
4   move 10 (⟨10⟩ − 1)
5   jump 2
6   jump 8
7   stop
8   (𝔼(print hello world) + 42)
9   (𝔼(jump 7) + 42)
10  jump 1
```

Cinematic

a_1 `inactivate` r_4
a_2 `move` r_4 2
a_3 `pop` r_4
a_4 `case` r_4
a_5 `jump` a_7
a_6 `jump` a_{11}
► r_1 a_7 `move` r_3 $\langle r_3 \mid 10 - \langle r_4 \mid 10 \rangle \rangle - 42$
a_8 `pop` r_3
a_9 `move` r_4 $\langle r_4 \mid 10 \rangle - 1$
a_{10} `pop` r_4
a_{11} `jump` a_4
a_{12} `activate` r_4
a_{13} `exec` r_4

r_2 a_{14} `stop`

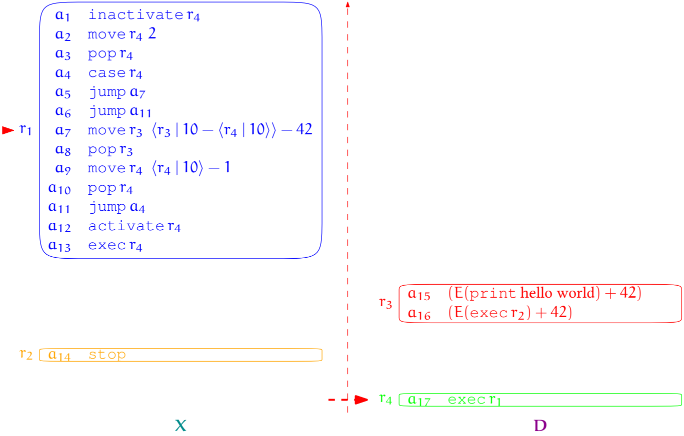
r_4 a_{17} `exec` r_1

X

r_3 a_{15} $(E($`print` hello world$) + 42)$
a_{16} $(E($`exec` $r_2) + 42)$

D

Cinematic



r_1

a_1 inactivate r_4
a_2 move r_4 2
a_3 pop r_4
a_4 case r_4
a_5 jump a_7
a_6 jump a_{11}
a_7 move r_3 $\langle r_3 \mid 10 - \langle r_4 \mid 10 \rangle\rangle - 42$
a_8 pop r_3
a_9 move r_4 $\langle r_4 \mid 10 \rangle - 1$
a_{10} pop r_4
a_{11} jump a_4
a_{12} activate r_4
a_{13} exec r_4

r_2 a_{14} stop

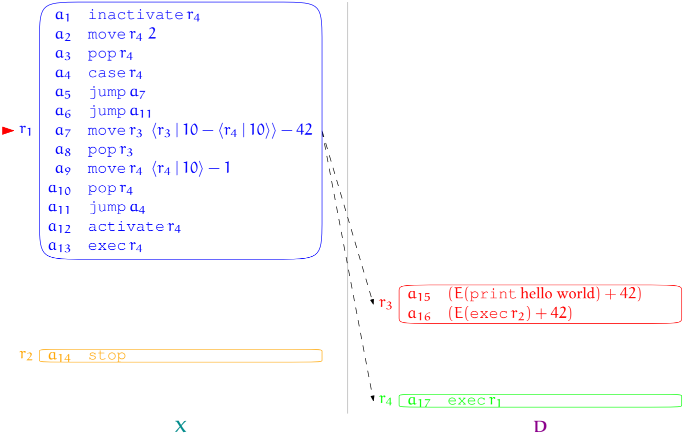
r_3
a_{15} $(E($print hello world$) + 42)$
a_{16} $(E($exec $r_2) + 42)$

r_4 a_{17} exec r_1

X

D

Cinematic



► r_1

a_1 `inactivate` r_4
a_2 `move` r_4 2
a_3 `pop` r_4
a_4 `case` r_4
a_5 `jump` a_7
a_6 `jump` a_{11}
a_7 `move` r_3 $\langle r_3 \mid 10 - \langle r_4 \mid 10 \rangle \rangle - 42$
a_8 `pop` r_3
a_9 `move` r_4 $\langle r_4 \mid 10 \rangle - 1$
a_{10} `pop` r_4
a_{11} `jump` a_4
a_{12} `activate` r_4
a_{13} `exec` r_4

r_2 a_{14} `stop`

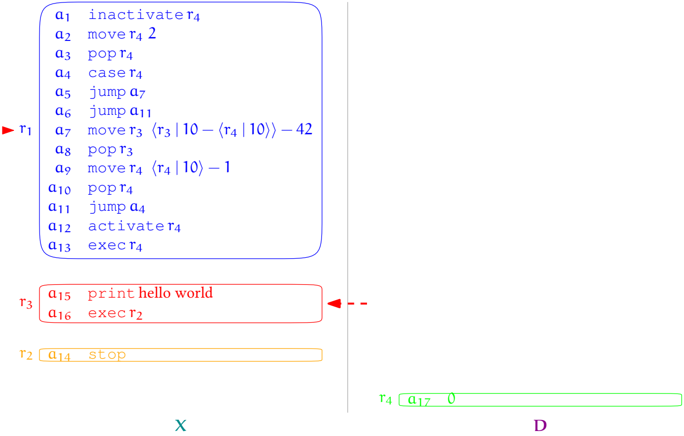
r_3
a_{15} $(E($`print` hello world$) + 42)$
a_{16} $(E($`exec` $r_2) + 42)$

r_4 a_{17} `exec` r_1

X

D

Cinematic



r_1

a_1 `inactivate` r_4
a_2 `move` r_4 2
a_3 `pop` r_4
a_4 `case` r_4
a_5 `jump` a_7
a_6 `jump` a_{11}
a_7 `move` r_3 $\langle r_3 \mid 10 - \langle r_4 \mid 10 \rangle \rangle - 42$
a_8 `pop` r_3
a_9 `move` r_4 $\langle r_4 \mid 10 \rangle - 1$
a_{10} `pop` r_4
a_{11} `jump` a_4
a_{12} `activate` r_4
a_{13} `exec` r_4

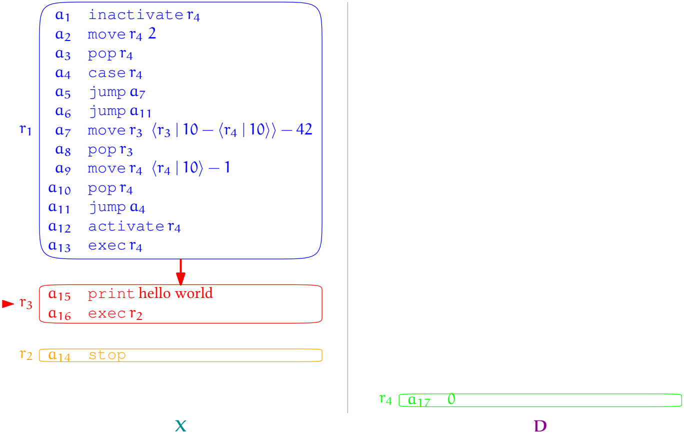
r_3
a_{15} `print` hello world
a_{16} `exec` r_2

r_2 a_{14} `stop`

r_4 a_{17} 0

X **D**

Cinematic



r_1

a_1 `inactivate` r_4

a_2 `move` r_4 2

a_3 `pop` r_4

a_4 `case` r_4

a_5 `jump` a_7

a_6 `jump` a_{11}

a_7 `move` r_3 $\langle r_3 \mid 10 - \langle r_4 \mid 10 \rangle \rangle - 42$

a_8 `pop` r_3

a_9 `move` r_4 $\langle r_4 \mid 10 \rangle - 1$

a_{10} `pop` r_4

a_{11} `jump` a_4

a_{12} `activate` r_4

a_{13} `exec` r_4

► r_3

a_{15} `print` hello world

a_{16} `exec` r_2

r_2

a_{14} `stop`

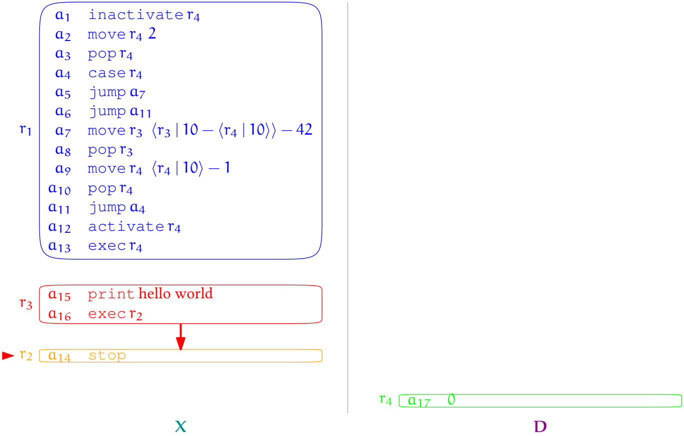
r_4

a_{17} 0

X

D

Cinematic



r_1

a_1 `inactivate` r_4

a_2 `move` r_4 2

a_3 `pop` r_4

a_4 `case` r_4

a_5 `jump` a_7

a_6 `jump` a_{11}

a_7 `move` r_3 $\langle r_3 \mid 10 - \langle r_4 \mid 10 \rangle\rangle - 42$

a_8 `pop` r_3

a_9 `move` r_4 $\langle r_4 \mid 10 \rangle - 1$

a_{10} `pop` r_4

a_{11} `jump` a_4

a_{12} `activate` r_4

a_{13} `exec` r_4

r_3

a_{15} `print` hello world

a_{16} `exec` r_2

► r_2

a_{14} `stop`

r_4

a_{17} 0

Computation

- Valuation $v_R \in \mathcal{V}$ with $R \subset \mathcal{R}$:

$$v_R = \{\langle r \rangle \mid r \in R\}$$

- State $s \in \mathcal{S}$:

$$(\underbrace{\mathtt{RP}, \mathtt{IP}, \mathbf{X}, v_{\mathbf{X}}}_{p \in \mathcal{P}}, v_{\mathbf{D}})$$

- Transition $\triangleright \in \wp(\mathcal{S}^2)$
- Interpretation of p (set of traces):

$$[\![p]\!] \overset{\text{def}}{=} \{s_1 \cdots s_n \in \mathcal{S}^* \mid s_1 = (p, v) \land \forall i \in [\![1, n-1]\!]\ s_i \triangleright s_{i+1}\}$$

Plan

Introduction

Framework

Applications

- Measure
- Program extraction
- Abstraction

Plan

Introduction

Framework

Applications

- Measure
- Program extraction
- Abstraction

Writing relation

Given a trace $\tau \in [\![p]\!]$

$$\begin{array}{cccccccccccccccccccccccccccc} 1 & \rightarrow & 2 & \rightarrow & 3 & \rightarrow & 4 & \rightarrow & 5 & \rightarrow & 2 & \rightarrow & 3 & \rightarrow & 4 & \rightarrow & 5 & \rightarrow & 2 & \rightarrow & 6 & \rightarrow & 8 & \rightarrow & 9 & \rightarrow & 5 \\ \mathit{1} & & \mathit{2} & & \mathit{3} & & \mathit{4} & & \mathit{5} & & \mathit{6} & & \mathit{7} & & \mathit{8} & & \mathit{9} & & \mathit{10} & & \mathit{11} & & \mathit{12} & & \mathit{13} & & \mathit{14} \end{array}$$

Writing relation: $\dashrightarrow_\tau \in \wp(\mathbb{N}^2)$

$i' \dashrightarrow_\tau i \iff$ step i' writes the code of an instruction which will be run at step i

Writing relation

Given a trace $\tau \in [\![p]\!]$

$$\begin{array}{cccccccccccccccccccccccccccc}
1 & \to & 2 & \to & 3 & \to & 4 & \to & 5 & \to & 2 & \to & 3 & \to & 4 & \to & 5 & \to & 2 & \to & 6 & \to & 8 & \to & 9 & \to & 5 \\
\mathit{1} & & \mathit{2} & & \mathit{3} & & \mathit{4} & & \mathit{5} & & \mathit{6} & & \mathit{7} & & \mathit{8} & & \mathit{9} & & \mathit{10} & & \mathit{11} & & \mathit{12} & & \mathit{13} & & \mathit{14}
\end{array}$$

Writing relation: $\dashrightarrow_\tau \in \wp(\mathbb{N}^2)$

$i' \dashrightarrow_\tau i \iff$ step i' writes the code of an instruction which will be run at step i

Example

Steps *2* and *11* write on address 5.

Writing relation

Given a trace $\tau \in [\![p]\!]$

$$\begin{tikzcd}[column sep=small]
1 \arrow[r] & 2 \arrow[r] \arrow[rrr, dashed, bend left] \arrow[rrrrrrr, dashed, bend left] & 3 \arrow[r] & 4 \arrow[r] & 5 \arrow[r] & 2 \arrow[r] & 3 \arrow[r] & 4 \arrow[r] & 5 \arrow[r] & 2 \arrow[r] & 6 \arrow[r] \arrow[rrr, dashed, bend left] & 8 \arrow[r] & 9 \arrow[r] & 5 \\
\mathit{1} & \mathit{2} & \mathit{3} & \mathit{4} & \mathit{5} & \mathit{6} & \mathit{7} & \mathit{8} & \mathit{9} & \mathit{10} & \mathit{11} & \mathit{12} & \mathit{13} & \mathit{14}
\end{tikzcd}$$

Writing relation: $\dashrightarrow_\tau \in \wp(\mathbb{N}^2)$

$i' \dashrightarrow_\tau i \quad \iff \quad$ step i' writes the code of an instruction which will be run at step i

Example

Steps *2* and *11* write on address 5.

Example

Program:

r_1	a_1	$\texttt{inactivate}\ r_4$
	a_2	$\texttt{move}\ r_4\ \ 2$
	a_3	$\texttt{pop}\ r_4$
	a_4	$\texttt{case}\ r_4$
	a_5	$\texttt{jump}\ a_7$
	a_6	$\texttt{jump}\ a_{11}$
	a_7	$\texttt{move}\ r_3\ \ \langle r_3 \mid 10 - \langle r_4 \mid 10 \rangle\rangle - 42$
	a_8	$\texttt{pop}\ r_3$
	a_9	$\texttt{move}\ r_4\ \ \langle r_4 \mid 10 \rangle - 1$
	a_{10}	$\texttt{pop}\ r_4$
	a_{11}	$\texttt{jump}\ a_4$
	a_{12}	$\texttt{activate}\ r_4$
	a_{13}	$\texttt{exec}\ r_4$
r_2	a_{14}	$\texttt{stop}$
r_3	a_{15}	$\mathbb{D}(\mathbb{E}(\texttt{print}\ \text{hello world}) + 42)$
	a_{16}	$\mathbb{D}(\mathbb{E}(\texttt{exec}\ r_2) + 42)$
r_4	a_{17}	$\texttt{exec}\ r_1$

Example

Program:

r_1	a_1	$\texttt{inactivate}\ r_4$
	a_2	$\texttt{move}\ r_4\ \ 2$
	a_3	$\texttt{pop}\ r_4$
	a_4	$\texttt{case}\ r_4$
	a_5	$\texttt{jump}\ a_7$
	a_6	$\texttt{jump}\ a_{11}$
	a_7	$\texttt{move}\ r_3\ \ \langle r_3 \mid 10 - \langle r_4 \mid 10\rangle\rangle - 42$
	a_8	$\texttt{pop}\ r_3$
	a_9	$\texttt{move}\ r_4\ \ \langle r_4 \mid 10\rangle - 1$
	a_{10}	$\texttt{pop}\ r_4$
	a_{11}	$\texttt{jump}\ a_4$
	a_{12}	$\texttt{activate}\ r_4$
	a_{13}	$\texttt{exec}\ r_4$
r_2	a_{14}	$\texttt{stop}$
r_3	a_{15}	$\mathbb{D}(\mathbb{E}(\texttt{print}\ \text{hello world}) + 42)$
	a_{16}	$\mathbb{D}(\mathbb{E}(\texttt{exec}\ r_2) + 42)$
r_4	a_{17}	$\texttt{exec}\ r_1$

Trace:

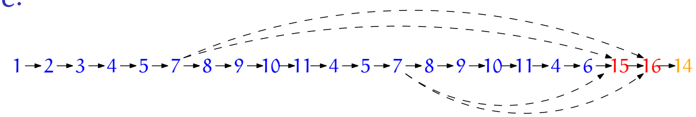
1 → 2 → 3 → 4 → 5 → 7 → 8 → 9 → 10 → 11 → 4 → 5 → 7 → 8 → 9 → 10 → 11 → 4 → 6 → 15 → 16 → 14

Example

Program:

r_1	a_1	$\texttt{inactivate}\ r_4$
	a_2	$\texttt{move}\ r_4\ \ 2$
	a_3	$\texttt{pop}\ r_4$
	a_4	$\texttt{case}\ r_4$
	a_5	$\texttt{jump}\ a_7$
	a_6	$\texttt{jump}\ a_{11}$
	a_7	$\texttt{move}\ r_3\ \ \langle r_3 \mid 10 - \langle r_4 \mid 10 \rangle\rangle - 42$
	a_8	$\texttt{pop}\ r_3$
	a_9	$\texttt{move}\ r_4\ \ \langle r_4 \mid 10 \rangle - 1$
	a_{10}	$\texttt{pop}\ r_4$
	a_{11}	$\texttt{jump}\ a_4$
	a_{12}	$\texttt{activate}\ r_4$
	a_{13}	$\texttt{exec}\ r_4$
r_2	a_{14}	$\texttt{stop}$
r_3	a_{15}	$\mathbb{D}(\mathbb{E}(\texttt{print}\ \text{hello world}) + 42)$
	a_{16}	$\mathbb{D}(\mathbb{E}(\texttt{exec}\ r_2) + 42)$
r_4	a_{17}	$\texttt{exec}\ r_1$

Trace:



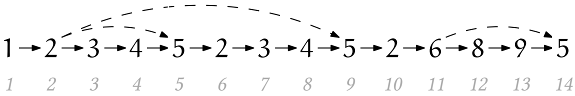
1 → 2 → 3 → 4 → 5 → 7 → 8 → 9 → 10 → 11 → 4 → 5 → 7 → 8 → 9 → 10 → 11 → 4 → 6 → 15 → 16 → 14

Trace interpretation

The monotone level $\eta_\tau(i)$:

> *“ The number of necessary self-modifications before executing step i”*

$$\begin{aligned}
\eta_\tau(1) &\overset{\text{def}}{=} 1 \\
\eta_\tau(i) &\overset{\text{def}}{=} \eta_\tau(i-1) \text{ if nobody wrote i} \\
\eta_\tau(i) &\overset{\text{def}}{=} \max\{\eta_\tau(i-1), \eta_\tau(i') + 1\} \text{ if } i' \dashrightarrow_\tau i.
\end{aligned}$$



Trace interpretation

The monotone level $\eta_\tau(i)$:

> *" The number of necessary self-modifications before executing step* i*"*

$$\begin{aligned} \eta_\tau(1) &\stackrel{\text{def}}{=} 1 \\ \eta_\tau(i) &\stackrel{\text{def}}{=} \eta_\tau(i-1) \text{ if nobody wrote i} \\ \eta_\tau(i) &\stackrel{\text{def}}{=} \max\{\eta_\tau(i-1), \eta_\tau(i') + 1\} \text{ if } i' \dashrightarrow_\tau i. \end{aligned}$$

$$1 \to 2 \to 3 \to 4 \to 5 \to 2 \to 3 \to 4 \to 5 \to 2 \to 6 \to 8 \to 9 \to 5$$

1 2 3 4 5 6 7 8 9 10 11 12 13 14

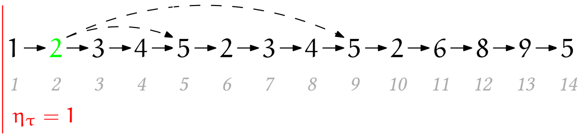
$\eta_\tau = 1$

Trace interpretation

The monotone level $\eta_\tau(i)$:

> “*The number of necessary self-modifications before executing step* i”

$$\begin{aligned}
\eta_\tau(1) &\overset{\text{def}}{=} 1 \\
\eta_\tau(i) &\overset{\text{def}}{=} \eta_\tau(i-1) \text{ if nobody wrote i} \\
\eta_\tau(i) &\overset{\text{def}}{=} \max\{\eta_\tau(i-1), \eta_\tau(i')+1\} \text{ if } i' \dashrightarrow_\tau i.
\end{aligned}$$

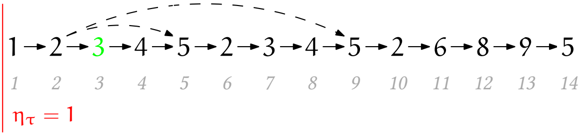


Trace interpretation

The monotone level $\eta_\tau(i)$:

> *“ The number of necessary self-modifications before executing step* i*”*

$$\begin{aligned}
\eta_\tau(1) &\overset{\text{def}}{=} 1\\
\eta_\tau(i) &\overset{\text{def}}{=} \eta_\tau(i-1) \text{ if nobody wrote i}\\
\eta_\tau(i) &\overset{\text{def}}{=} \max\{\eta_\tau(i-1), \eta_\tau(i')+1\} \text{ if } i' \dashrightarrow_\tau i.
\end{aligned}$$

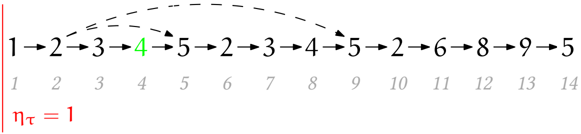


Trace interpretation

The monotone level $\eta_\tau(i)$:

> *" The number of necessary self-modifications before executing step* i*"*

$$\begin{aligned}
\eta_\tau(1) &\overset{\text{def}}{=} 1\\
\eta_\tau(i) &\overset{\text{def}}{=} \eta_\tau(i-1) \text{ if nobody wrote i}\\
\eta_\tau(i) &\overset{\text{def}}{=} \max\{\eta_\tau(i-1), \eta_\tau(i')+1\} \text{ if } i' \dashrightarrow_\tau i.
\end{aligned}$$



Trace interpretation

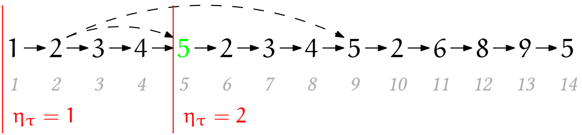
The monotone level $\eta_\tau(i)$:

> *“ The number of necessary self-modifications before executing step* i*”*

$$\eta_\tau(1) \overset{\text{def}}{=} 1$$

$$\eta_\tau(i) \overset{\text{def}}{=} \eta_\tau(i-1) \text{ if nobody wrote i}$$

$$\eta_\tau(i) \overset{\text{def}}{=} \max\{\eta_\tau(i-1), \eta_\tau(i') + 1\} \text{ if } i' \dashrightarrow_\tau i.$$

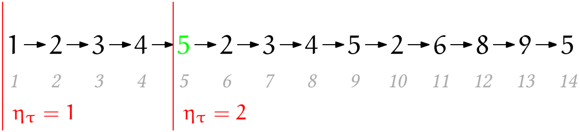


Trace interpretation

The monotone level $\eta_\tau(i)$:

> *" The number of necessary self-modifications before executing step* i*"*

$$
\begin{aligned}
\eta_\tau(1) &\overset{\text{def}}{=} 1\\
\eta_\tau(i) &\overset{\text{def}}{=} \eta_\tau(i-1) \text{ if nobody wrote i}\\
\eta_\tau(i) &\overset{\text{def}}{=} \max\{\eta_\tau(i-1), \eta_\tau(i')+1\} \text{ if } i' \dashrightarrow_\tau i.
\end{aligned}
$$



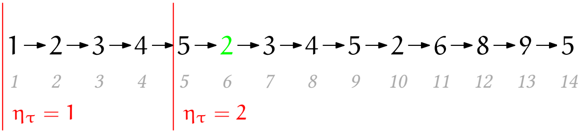
$$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 2 \rightarrow 6 \rightarrow 8 \rightarrow 9 \rightarrow 5$$

Trace interpretation

The monotone level $\eta_\tau(i)$:

> *“ The number of necessary self-modifications before executing step* i*”*

$$\begin{aligned}
\eta_\tau(1) &\overset{\text{def}}{=} 1\\
\eta_\tau(i) &\overset{\text{def}}{=} \eta_\tau(i-1) \text{ if nobody wrote i}\\
\eta_\tau(i) &\overset{\text{def}}{=} \max\{\eta_\tau(i-1), \eta_\tau(i')+1\} \text{ if } i' \dashrightarrow_\tau i.
\end{aligned}$$



$1 \to 2 \to 3 \to 4 \to 5 \to 2 \to 3 \to 4 \to 5 \to 2 \to 6 \to 8 \to 9 \to 5$

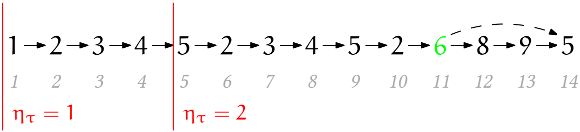
$\eta_\tau = 1$ $\eta_\tau = 2$

Trace interpretation

The monotone level $\eta_\tau(i)$:

> *“ The number of necessary self-modifications before executing step* i*”*

$$\begin{aligned}
\eta_\tau(1) &\overset{\text{def}}{=} 1 \\
\eta_\tau(i) &\overset{\text{def}}{=} \eta_\tau(i-1) \text{ if nobody wrote i} \\
\eta_\tau(i) &\overset{\text{def}}{=} \max\{\eta_\tau(i-1), \eta_\tau(i') + 1\} \text{ if } i' \dashrightarrow_\tau i.
\end{aligned}$$



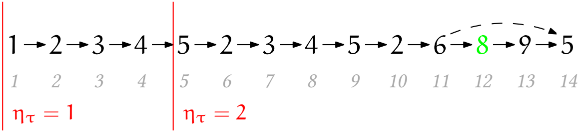
$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 2 \rightarrow 6 \rightarrow 8 \rightarrow 9 \rightarrow 5$

Trace interpretation

The monotone level $\eta_\tau(i)$:

> *“ The number of necessary self-modifications before executing step* i*”*

$$
\begin{aligned}
\eta_\tau(1) &\overset{\text{def}}{=} 1\\
\eta_\tau(i) &\overset{\text{def}}{=} \eta_\tau(i-1) \text{ if nobody wrote i}\\
\eta_\tau(i) &\overset{\text{def}}{=} \max\{\eta_\tau(i-1), \eta_\tau(i')+1\} \text{ if } i' \dashrightarrow_\tau i.
\end{aligned}
$$

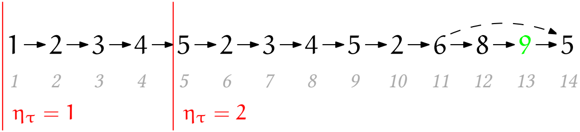


Trace interpretation

The monotone level $\eta_\tau(i)$:

> *“The number of necessary self-modifications before executing step* i*”*

$$
\begin{aligned}
\eta_\tau(1) &\overset{\text{def}}{=} 1 \\
\eta_\tau(i) &\overset{\text{def}}{=} \eta_\tau(i-1) \text{ if nobody wrote i} \\
\eta_\tau(i) &\overset{\text{def}}{=} \max\{\eta_\tau(i-1), \eta_\tau(i')+1\} \text{ if } i' \dashrightarrow_\tau i.
\end{aligned}
$$



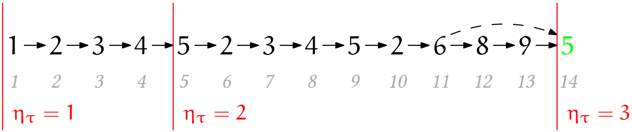
$1 \to 2 \to 3 \to 4 \to 5 \to 2 \to 3 \to 4 \to 5 \to 2 \to 6 \to 8 \to 9 \to 5$

Trace interpretation

The monotone level $\eta_\tau(i)$:

> *“The number of necessary self-modifications before executing step* i*”*

$$\begin{aligned}
\eta_\tau(1) &\overset{\text{def}}{=} 1\\
\eta_\tau(i) &\overset{\text{def}}{=} \eta_\tau(i-1) \text{ if nobody wrote i}\\
\eta_\tau(i) &\overset{\text{def}}{=} \max\{\eta_\tau(i-1), \eta_\tau(i')+1\} \text{ if } i' \dashrightarrow_\tau i.
\end{aligned}$$

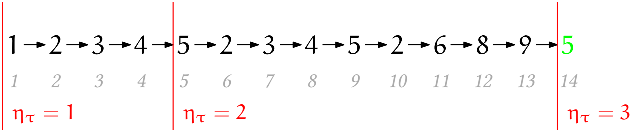


Trace interpretation

The monotone level $\eta_\tau(i)$:

> *“ The number of necessary self-modifications before executing step* i*”*

$$\begin{aligned} \eta_\tau(1) &\overset{\text{def}}{=} 1 \\ \eta_\tau(i) &\overset{\text{def}}{=} \eta_\tau(i-1) \text{ if nobody wrote i} \\ \eta_\tau(i) &\overset{\text{def}}{=} \max\{\eta_\tau(i-1), \eta_\tau(i') + 1\} \text{ if } i' \dashrightarrow_\tau i. \end{aligned}$$



$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 2 \rightarrow 6 \rightarrow 8 \rightarrow 9 \rightarrow 5$

1 2 3 4 5 6 7 8 9 10 11 12 13 14

$\eta_\tau = 1$ $\eta_\tau = 2$ $\eta_\tau = 3$

Waves

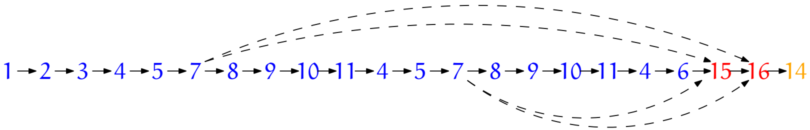
A wave of level n:

$$w_n \overset{\text{def}}{=} \{i \mid \eta_\tau(i) = n\}$$

Waves

A wave of level n:

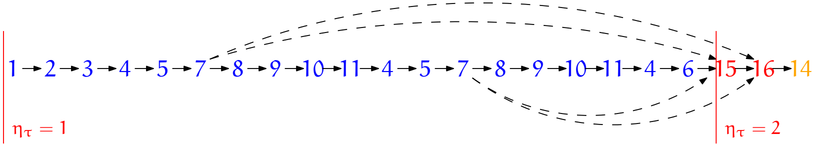
$$w_n \overset{\text{def}}{=} \{i \mid \eta_\tau(i) = n\}$$



Waves

A wave of level n:

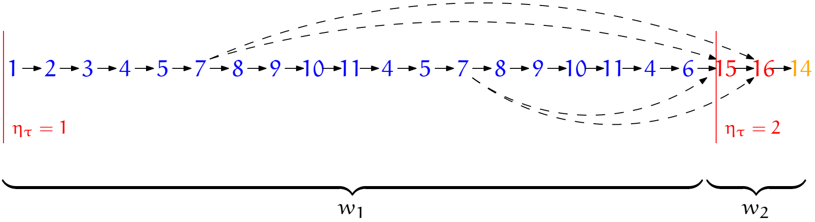
$$w_n \overset{\text{def}}{=} \{i \mid \eta_\tau(i) = n\}$$



Waves

A wave of level n:

$$w_n \overset{\text{def}}{=} \{i \mid \eta_\tau(i) = n\}$$



1 → 2 → 3 → 4 → 5 → 7 → 8 → 9 → 10 → 11 → 4 → 5 → 7 → 8 → 9 → 10 → 11 → 4 → 6 → 15 → 16 → 14

$\eta_\tau = 1$

$\eta_\tau = 2$

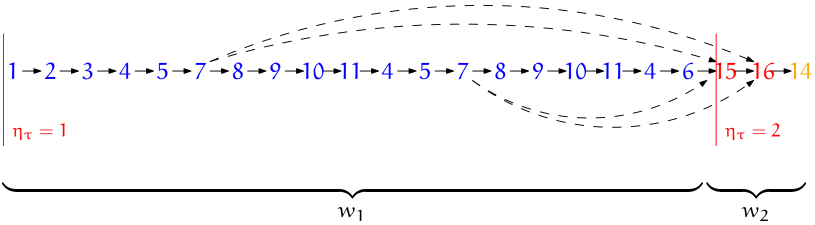
w_1

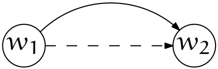
w_2

Waves

A wave of level n:

$$w_n \overset{\text{def}}{=} \{i \mid \eta_\tau(i) = n\}$$





Sum up

- Measure self-modification.
  - → the writing relation.
- Interpretation wrt self-modification
  - → the trace-oriented notion of wave.

Sum up

- Measure self-modification.
  - → the writing relation.
- Interpretation wrt self-modification
  - → the trace-oriented notion of wave.

Question

Reconstruct non self-modifying program?

Plan

Introduction

Framework

Applications

- Measure
- Program extraction
- Abstraction

Intuition

Question

How do I switch from a wave to another?

Intuition

Question

How do I switch from a wave to another?

Answer

When I execute something I wrote.

Intuition

Question

How do I switch from a wave to another?

Answer

When I execute something I wrote.

ASM_2 answer

When I execute a register which was not activated when I began.

→ A good witness of a wave is thus **X** when the wave begins.

Witness

Given a trace $\tau = s_1 \cdots s_n$, a wave $w = [\![i, j]\!]$ of τ, the witness of w is

$$\operatorname{prog}_\tau w \stackrel{\text{def}}{=} p_{\min w} \text{ where } \forall i, s_i = (p_i, v_{\mathbf{D}_i})$$

$\operatorname{prog}_\tau w$ is a snapshot of the executable memory at the beginning of w.

Witness

Given a trace $\tau = s_1 \cdots s_n$, a wave $w = [\![i, j]\!]$ of τ, the witness of w is

$$\mathrm{prog}_\tau\, w \overset{\text{def}}{=} p_{\min w} \text{ where } \forall i, s_i = (p_i, \nu_{\mathbf{D}_i})$$

$\mathrm{prog}_\tau\, w$ is a snapshot of the executable memory at the beginning of w.

Soundness

$$s_i \cdots s_j \in [\![\mathrm{prog}_\tau(w)]\!] \,.$$

Idea

All registers executed in w are already present in $\mathrm{prog}_\tau(w)$: otherwise a register have to be activated (so written) and we change of wave when it is executed.

Plan

Introduction

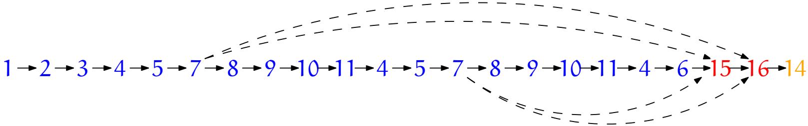
Framework

Applications

- Measure
- Program extraction
- Abstraction

Abstract execution

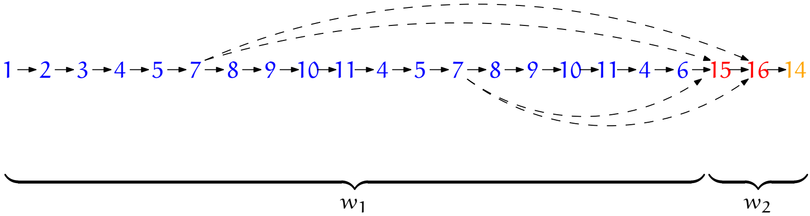
Given a trace $\tau \in [\![p]\!]$



1 → 2 → 3 → 4 → 5 → 7 → 8 → 9 → 10 → 11 → 4 → 5 → 7 → 8 → 9 → 10 → 11 → 4 → 6 → 15 → 16 → 14

Abstract execution

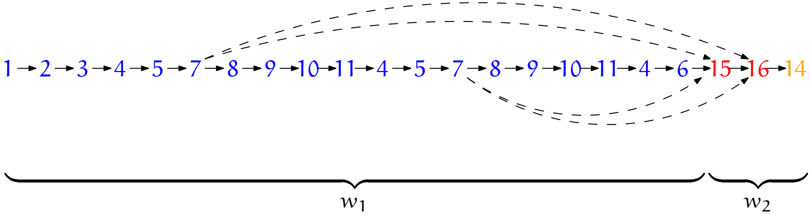
Given a trace $\tau \in [\![p]\!]$



$$\underbrace{1 \to 2 \to 3 \to 4 \to 5 \to 7 \to 8 \to 9 \to 10 \to 11 \to 4 \to 5 \to 7 \to 8 \to 9 \to 10 \to 11 \to 4 \to 6}_{w_1} \to \underbrace{15 \to 16 \to 14}_{w_2}$$

Abstract execution

Given a trace $\tau \in [\![p]\!]$



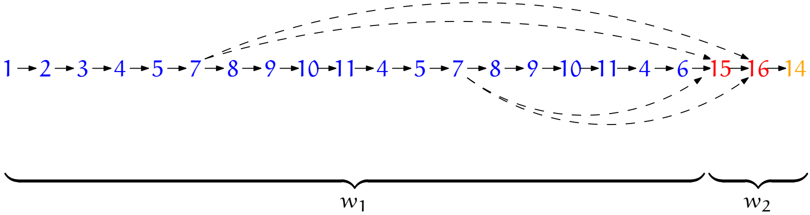
$$\underbrace{1 \to 2 \to 3 \to 4 \to 5 \to 7 \to 8 \to 9 \to 10 \to 11 \to 4 \to 5 \to 7 \to 8 \to 9 \to 10 \to 11 \to 4 \to 6}_{w_1} \to \underbrace{15 \to 16 \to 14}_{w_2}$$

we group steps into waves:

$$w_1 \cdot w_2$$

Abstract execution

Given a trace $\tau \in [\![p]\!]$



we group steps into waves:

$$w_1 \cdot w_2$$

and recover program for each wave:

$$p_1 \cdot p_2.$$

This is the abstract execution $\alpha(\tau)$.

Abstract semantics

Construct a correct abstract semantics wrt α:

Abstract semantics

Construct a correct abstract semantics wrt α:
Abstract semantics of p

$$[\![p]\!]^{\#} \quad : \quad \text{set of sequences of programs}$$

Abstract semantics

Construct a correct abstract semantics wrt α:

Abstract semantics of p

$$\begin{array}{ccl} [\![\mathrm{p}]\!]^{\#} & : & \text{set of sequences of programs} \\ \triangleright^{\#} & : & \text{transition function in } \wp(\mathcal{P}^2) \end{array}$$

Abstract semantics

Construct a correct abstract semantics wrt α:
Abstract semantics of p

$$\begin{array}{rcl} [\![p]\!]^\# & : & \text{set of sequences of programs} \\ \triangleright^\# & : & \text{transition function in } \wp(\mathcal{P}^2) \end{array}$$

such that

$$\alpha([\![p]\!]) \subseteq [\![p]\!]^\#$$

Abstract semantics

Abstract transition $\triangleright^{\#} \in \wp(\mathcal{P}^{\mathcal{P}})$:

$p \triangleright^{\#} p' \iff p'$ is the witness of the 2^{nd} wave of a $\tau \in [\![p]\!]$

Abstract interpretation $[\![p]\!]^{\#}$:

$$[\![p]\!]^{\#} \overset{\text{def}}{=} \{p_1 \cdots p_n \in \mathcal{P}^* \mid p_1 = p \land \forall i \in [\![1, n-1]\!], p_i \triangleright^{\#} p_{i+1}\}$$

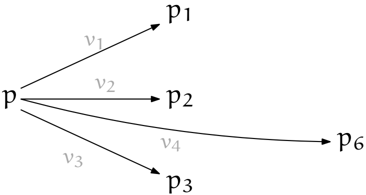
Example: $[\![p]\!]^\#$

Valuations $v_1, v_2, v_3, v_4 \in \mathcal{V}$

p

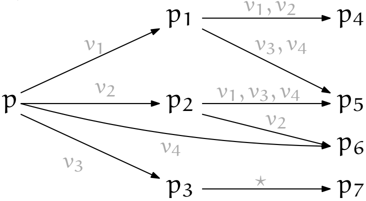
Example: $[\![\mathrm{p}]\!]^{\#}$

Valuations $v_1, v_2, v_3, v_4 \in \mathcal{V}$



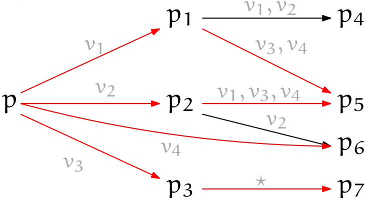
Example: $[\![p]\!]^{\#}$

Valuations $v_1, v_2, v_3, v_4 \in \mathcal{V}$



Example: $[\![\mathrm{p}]\!]^{\#}$

Valuations $v_1, v_2, v_3, v_4 \in \mathcal{V}$



$\alpha([\![\mathrm{p}]\!]) \subseteq [\![\mathrm{p}]\!]^{\#}$

Conclusion

We have…

- built abstract machine for self-modification,
- extracted non self-modifying programs for each waves,
- constructed abstract views from self-modifying programs.

Conclusion

We have…

- built abstract machine for self-modification,
- extracted non self-modifying programs for each waves,
- constructed abstract views from self-modifying programs.

We will…

- define non monotone waves,
- improve symmetry of the definition (read/write),
- take advantage of intermediate granularity.

Bibliography

M. D. Preda, R. Giacobazzi, and S. Debray.
Modeling metamorphism by abstract interpretation.
Theoretical Computer Science, 2012.

D. Reynaud.
Analyse de codes auto-modifiants pour la sécurité informatique.
PhD thesis, INPL, 2010.

C. C. Elgot and A. Robinson.
Random-access stored-program machines, an approach to programming languages.
Journal of the Association for Computing Machinery, 11(4):365–399, October 1964.

C.-K. Hur and D. Dreyer.
A kripke logical relation between ml and assembly.
Principles of programming languages, 2011.

Annexe: non monotone waves

The set of waves:

$$\mathcal{W}_\tau \overset{\text{def}}{=} \mathbb{N}/_{\sim_\tau}$$

The wave relation $\sim_\tau$:

- $\star \dashrightarrow_\tau i$ if i is written by nobody
- $\star \sim_\tau \star$
- $i \sim_\tau j \iff \exists i' \sim_\tau j', i' \dashrightarrow_\tau i \wedge j' \dashrightarrow_\tau j$

Properties

$\sim_\tau$ is an equivalence relation.

Annexe: abstract interpretation

$$\triangleright^{\#}\, p \overset{\text{def}}{=} \{p' | \exists\, \tau = s_1 \cdots s_{n+1} \in [\![p]\!]\,, p' = \mathrm{prog}(s_{n+1}) \wedge \forall i, j \in [\![1, n]\!]\,, i \sim_{\tau} j\}$$

$$\mathsf{T}^{\#}\, \mathsf{X} \overset{\text{def}}{=} \mathsf{X} \cup \{p_1 \cdots p_n p_{n+1} \mid p_1 \cdots p_n \in \mathsf{X} \wedge \triangleright^{\#}\, p_n = p_{n+1}\}$$

$$[\![p]\!]^{\#} \overset{\text{def}}{=} \mathsf{Fix}_{\{p\}}\, \mathsf{T}^{\#}$$