On the Nature of Symbolic Execution[1]

Frank de Boer (Joint work with Marcello Bonsangue)

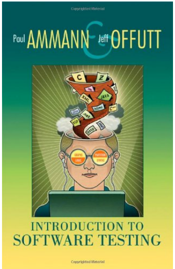
IFIP WG 2.2, 2019

[1] FM 2019

Motivation: No Formal Theory

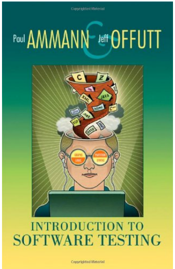
Motivation: No Formal Theory

Master course at Leiden University



Motivation: No Formal Theory

Master course at Leiden University



Tools

- No formal specification (of correctness/completeness)

Basic Symbolic Execution

Basic Symbolic Execution

Programming expressions

$$e \quad := \quad x \mid op(e_1, \ldots, e_n)$$

where x is a *simple* variable of a *basic* type.

Basic Symbolic Execution

Programming expressions

$$e \quad := \quad x \mid op(e_1, \ldots, e_n)$$

where x is a *simple* variable of a *basic* type.

Substitution $\sigma : \mathit{Var} \rightarrow \mathit{Expr}$

$$\begin{aligned} x\sigma &= \sigma(x) \\ op(e_1, \ldots, e_n)\sigma &= op(e_1\sigma, \ldots, e_n\sigma) \end{aligned}$$

Basic Symbolic Execution

Programming expressions

$$e \quad := \quad x \mid op(e_1, \ldots, e_n)$$

where x is a *simple* variable of a *basic* type.

Substitution $\sigma : Var \rightarrow Expr$

$$\begin{aligned} x\sigma &= \sigma(x) \\ op(e_1, \ldots, e_n)\sigma &= op(e_1\sigma, \ldots, e_n\sigma) \end{aligned}$$

Symbolic configuration

$\langle S, \sigma, \phi \rangle$ where

- S denotes the statement to be executed,
- σ denotes the current substitution,
- Boolean condition ϕ denotes the path condition.

Symbolic Transition System

Symbolic Transition System

Assignment

$$\langle x = e; S, \sigma, \phi \rangle \rightarrow \langle S, \sigma[x = e\sigma], \phi \rangle$$

where $\sigma[x = e](y) = \sigma(y)$ if x and y are distinct variables, and $\sigma[x = e](x) = e$ otherwise.

Symbolic Transition System

Assignment

$$\langle x = e; S, \sigma, \phi \rangle \rightarrow \langle S, \sigma[x = e\sigma], \phi \rangle$$

where $\sigma[x = e](y) = \sigma(y)$ if x and y are distinct variables, and $\sigma[x = e](x) = e$ otherwise.

Choice

- $\langle \mathit{if}\ B\ \{S_1\}\{S_2\}; S, \sigma, \phi \rangle \rightarrow \langle S_1; S, \sigma, \phi \wedge B\sigma \rangle$
- $\langle \mathit{if}\ B\ \{S_1\}\{S_2\}; S, \sigma, \phi \rangle \rightarrow \langle S_2; S, \sigma, \phi \wedge \neg B\sigma \rangle$

Symbolic Transition System

Assignment

$$\langle x = e; S, \sigma, \phi \rangle \rightarrow \langle S, \sigma[x = e\sigma], \phi \rangle$$

where $\sigma[x = e](y) = \sigma(y)$ if x and y are distinct variables, and $\sigma[x = e](x) = e$ otherwise.

Choice

- $\langle \textit{if } B\ \{S_1\}\{S_2\}; S, \sigma, \phi \rangle \rightarrow \langle S_1; S, \sigma, \phi \wedge B\sigma \rangle$
- $\langle \textit{if } B\ \{S_1\}\{S_2\}; S, \sigma, \phi \rangle \rightarrow \langle S_2; S, \sigma, \phi \wedge \neg B\sigma \rangle$

Iteration

- $\langle \textit{while } B\ \{S\}; S', \sigma, \phi \rangle \rightarrow \langle S; \textit{while } B\ \{S\}; S', \sigma, \phi \wedge B\sigma \rangle$
- $\langle \textit{while } B\ \{S\}; S', \sigma, \phi \rangle \rightarrow \langle S', \sigma, \phi \wedge \neg B\sigma \rangle$

Correctness

Correctness

Concrete transitions

$$\langle S, V \rangle \rightarrow \langle S', V' \rangle$$

where $V : \mathit{Var} \rightarrow \mathit{Val}$

Correctness

Concrete transitions

$$\langle S, V\rangle \rightarrow \langle S', V'\rangle$$

where $V : Var \rightarrow Val$

Theorem

If $\langle S, id, true\rangle \rightarrow^* \langle S', \sigma, \phi\rangle$ and $V(\phi) = \mathrm{true}$ then

$$\langle S, V\rangle \rightarrow^* \langle S', V \circ \sigma\rangle$$

where $V \circ \sigma(x) = V(\sigma(x))$.

Completeness

Completeness

Relating symbolic and concrete configurations

$$\langle S, V\rangle \simeq \langle S, \sigma, \phi\rangle$$

if $V = V_0 \circ \sigma$ and $V_0(\phi) = \mathit{true}$, for some valuation V_0.

Completeness

Relating symbolic and concrete configurations

$$\langle S, V\rangle \simeq \langle S, \sigma, \phi\rangle$$

if $V = V_0 \circ \sigma$ and $V_0(\phi) = \textit{true}$, for some valuation V_0.

Theorem (simulation)

$$\langle S, V\rangle \simeq \langle S, \sigma, \phi\rangle \text{ and } \langle S, V\rangle \rightarrow \langle S', V'\rangle$$

implies the existence of a corresponding symbolic transition

$$\langle S, \sigma, \phi\rangle \rightarrow \langle S', \sigma', \phi'\rangle$$

such that $\langle S', V'\rangle \simeq \langle S', \sigma', \phi'\rangle$.

OO

OO

Variables

- Global variables (main statement)
- Local variables (formal parameters of methods)
- Instance variables (class definitions)

OO

Variables

- Global variables (main statement)
- Local variables (formal parameters of methods)
- Instance variables (class definitions)

Programming expressions

$$e \quad := \quad x \mid op(e_1, \ldots, e_n)$$

(In the main statement only global variables are used).

OO

Variables

- Global variables (main statement)
- Local variables (formal parameters of methods)
- Instance variables (class definitions)

Programming expressions

$$e \quad := \quad x \mid op(e_1, \ldots, e_n)$$

(In the main statement only global variables are used).

Syntax of *heap variables H* and *heap expressions E*

$$\begin{array}{lcl} H & := & x \mid H.y \\ E & := & H \mid op(E_1, \ldots, E_n)\,, \end{array}$$

where x is a global variable.

Symbolic Heap Representation

Symbolic Heap Representation

Symbolic heap σ

σ denotes a substitution which assigns to each heap variable H a heap expression E.

Symbolic Heap Representation

Symbolic heap σ

σ denotes a substitution which assigns to each heap variable H a heap expression E.

Local environment τ

τ denotes a substitution which assigns to each formal parameter x a heap expression E.

Symbolic Heap Representation

Symbolic heap σ

σ denotes a substitution which assigns to each heap variable H a heap expression E.

Local environment τ

τ denotes a substitution which assigns to each formal parameter x a heap expression E.

Application substitution $\theta = \tau \cup \sigma$

$x\theta$	$= \sigma(x)$	global variable
$x\theta$	$= \tau(x)$	local variable
$x\theta$	$= \sigma(\tau(this).x)$	instance variable
$op(E_1, \ldots, E_n)\theta$	$= op(E_1\theta, \ldots, E_n\theta)$	

Symbolic Heap Update

Symbolic Heap Update

Update global variable

- $\sigma[x = E](x) = E$
- $\sigma[x = E](H) = \sigma(H)$, for any other heap variable H

Symbolic Heap Update

Update global variable

- $\sigma[x = E](x) = E$
- $\sigma[x = E](H) = \sigma(H)$, for any other heap variable H

Update instance variable

- $\sigma[H.x = E](H'.x) = \textit{if}\ \sigma(H') = \sigma(H)\ \textit{then}\ E\ \textit{else}\ \sigma(H'.x)\ \textit{fi}$
- $\sigma[H.x = E](H') = \sigma(H')$, for any other heap variable H'

Symbolic Transition System

Symbolic Transition System

Assignment global variable

$$\langle(\bot, x = e; S), \sigma, \phi\rangle \rightarrow \langle(\bot, S), \sigma[x = e\sigma], \phi\rangle$$

Symbolic Transition System

Assignment global variable

$$\langle(\perp, x = e; S), \sigma, \phi\rangle \rightarrow \langle(\perp, S), \sigma[x = e\sigma], \phi\rangle$$

Assignment instance variable

$$\langle(\tau, x = e; S) \cdot \Sigma, \sigma, \phi\rangle \rightarrow \langle(\tau, S) \cdot \Sigma, \sigma[\tau(\mathit{this}).x = e\theta], \phi\rangle$$

where $\theta = \tau \cup \sigma$.

Symbolic Transition System

Assignment global variable

$$\langle (\bot, x = e; S), \sigma, \phi \rangle \rightarrow \langle (\bot, S), \sigma[x = e\sigma], \phi \rangle$$

Assignment instance variable

$$\langle (\tau, x = e; S) \cdot \Sigma, \sigma, \phi \rangle \rightarrow \langle (\tau, S) \cdot \Sigma, \sigma[\tau(\mathit{this}).x = e\theta], \phi \rangle$$

where $\theta = \tau \cup \sigma$.

Object creation

$$\langle (\tau, x = \mathrm{new}\ C; S) \cdot \Sigma, \sigma, \phi \rangle \rightarrow \langle (\tau[x = y], S) \cdot \Sigma, \sigma', \phi \rangle$$

where $\sigma'(y.z) = \mathrm{nil}$.

Symbolic Transition System (Cont'd)

Symbolic Transition System (Cont'd)

Method call

Given a method declaration $m(\bar{u})\{S\}$, we have

$$\langle(\tau, y = e_0.m(\bar{e}); S') \cdot \Sigma, \sigma, \phi\rangle \rightarrow \langle(\tau'.S) \cdot (\tau, y =?; S') \cdot \Sigma, \sigma, \phi'\rangle$$

where

- $\tau'(\bar{u}) = \bar{e}(\tau \cup \sigma)$
- $\tau'(this) = e_0(\tau \cup \sigma)$

Symbolic Transition System (Cont'd)

Method call

Given a method declaration $m(\bar{u})\{S\}$, we have

$$\langle(\tau, y = e_0.m(\bar{e}); S') \cdot \Sigma, \sigma, \phi\rangle \rightarrow \langle(\tau'.S) \cdot (\tau, y =?; S') \cdot \Sigma, \sigma, \phi'\rangle$$

where

- $\tau'(\bar{u}) = \bar{e}(\tau \cup \sigma)$
- $\tau'(this) = e_0(\tau \cup \sigma)$

Method return

$$\langle(\tau, \text{return } e) \cdot (\tau', x =?; S) \cdot \Sigma, \sigma, \phi\rangle \rightarrow \langle(\tau'[x = e\theta], S) \cdot \Sigma, \sigma, \phi\rangle$$

where $\theta = (\tau \cup \sigma)$.

Concrete Transition System

Concrete Transition System

Valuation

- $V(H) = V(H')$ implies $V(H.x) = V(H'.x)$,
- $V(x) \neq V(x')$,
  for any two distinct global variables x and x' which do not appear in the main statement (*unique name assumption*).

Concrete Transition System

Valuation

- $V(H) = V(H')$ implies $V(H.x) = V(H'.x)$,
- $V(x) \neq V(x')$,
  for any two distinct global variables x and x' which do not appear in the main statement (*unique name assumption*).

Heap update

- $V[H.x = v](H'.x) = \begin{cases} v & \text{if } V(H') = V(H) \\ V(H'.x) & \text{otherwise} \end{cases}$
- $V[H.x = v](H') = V(H')$, for any other heap variable H'.

Concrete Transition System

Valuation

- $V(H) = V(H')$ implies $V(H.x) = V(H'.x)$,
- $V(x) \neq V(x')$,
  for any two distinct global variables x and x' which do not appear in the main statement (*unique name assumption*).

Heap update

- $V[H.x = v](H'.x) = \begin{cases} v & \text{if } V(H') = V(H) \\ V(H'.x) & \text{otherwise} \end{cases}$
- $V[H.x = v](H') = V(H')$, for any other heap variable H'.

Assignment instance variable

$$\langle (L, x = e; S) \cdot \Sigma, V \rangle \rightarrow \langle (L, S) \cdot \Sigma, V[H.x = v] \rangle$$

where $V(H) = L(\mathit{this})$ and $v = (L \cup V)(e)$.

Correctness

Correctness

Theorem

If $\langle (\bot, S), id, \mathrm{true} \rangle \to^* \langle (\tau, S') \cdot \Sigma, \sigma, \phi \rangle$ and $V(\phi) = \mathrm{true}$, where V is an initial valuation, then

$$\langle (\bot, S), V \rangle \to^* \langle (V \circ \tau, S') \cdot V \circ \Sigma, V \circ \sigma \rangle$$

where

- $(V \circ \tau)(x) = V(\tau(x))$
- $(V \circ \sigma)(H) = V(\sigma(H))$

Conclusion

Conclusion

Extensions

- Arrays
- Multithreading (Java)
- Concurrent objects
- Concolic execution
- Backward symbolic execution

Conclusion

Extensions

- Arrays
- Multithreading (Java)
- Concurrent objects
- Concolic execution
- Backward symbolic execution

Implementation

- Finite representation
- Optimization aliasing

Conclusion

Extensions

- Arrays
- Multithreading (Java)
- Concurrent objects
- Concolic execution
- Backward symbolic execution

Implementation

- Finite representation
- Optimization aliasing

Related work

A generic framework for symbolic execution: A coinductive approach.
By D. Lucanu, V. Rusu, and A. Arusoaie.
In Journal of Symbolic Computation 80(1):125–163, Elsevier,