Semantics of Symbolic Execution

Arie Gurfinkel

February 15, 2019

UNIVERSITY OF WATERLOO

Preliminaries

- L is a set of program variables (locations)
- *concrete* state q (a.k.a., an *environment*) is a map from program variables L to integers $\mathbb{Z}$
- $Q : L \to \mathbb{Z}$ is the set of all states.

Notation

- $q(v)$ stands for the value of variable v in state q
- $[\,]$ stands for an empty state (no variables)
- $[x := u, y := v]$ stands for a state s.t. x is u and y is v
- $q[x := n]$ is a state obtained from q by sub the value of x by n

UNIVERSITY OF WATERLOO

Operational Semantics: Expressions

$$\frac{}{\langle n, q\rangle \Downarrow n}\, n \in \mathbb{Z} \qquad \frac{}{\langle v, q\rangle \Downarrow q(v)}\, v \in L$$

$$\frac{\langle a_1, q\rangle \Downarrow n_1 \qquad \langle a_2, q\rangle \Downarrow n_2}{\langle a_1 + a_2, q\rangle \Downarrow n_1 + n_2} \qquad \frac{\langle a_1, q\rangle \Downarrow n_1 \qquad \langle a_2, q\rangle \Downarrow n_2}{\langle a_1 \times a_2, q\rangle \Downarrow n_1 \times n_2}$$

$$\frac{\langle a_1, q\rangle \Downarrow n_1 \qquad \langle a_2, q\rangle \Downarrow n_2}{\langle a_1 < a_2, q\rangle \Downarrow n_1 < n_2} \qquad \frac{\langle a_1, q\rangle \Downarrow n_1 \qquad \langle a_2, q\rangle \Downarrow n_2}{\langle a_1 > a_2, q\rangle \Downarrow n_1 > n_2}$$

$$\frac{\langle b_1, q\rangle \Downarrow r_1 \qquad \langle b_2, q\rangle \Downarrow r_2}{\langle b_1 \wedge b_2, q\rangle \Downarrow r_1 \wedge r_2} \qquad \frac{\langle b_1, q\rangle \Downarrow r_1 \qquad \langle b_2, q\rangle \Downarrow r_2}{\langle b_1 \vee b_2, q\rangle \Downarrow r_1 \vee r_2}$$

UNIVERSITY OF WATERLOO

Operational Semantics: Statements

$$\frac{}{\langle \textbf{skip}, q\rangle \Downarrow q}$$

$$\frac{}{\langle \textbf{print\_state}, q\rangle \Downarrow q}$$

$$\frac{\langle s_1, q\rangle \Downarrow q'' \qquad \langle s_2, q''\rangle \Downarrow q'}{\langle s_1 \,;\, s_2, q\rangle \Downarrow q'}$$

$$\frac{\langle e, q\rangle \Downarrow n}{\langle x := e, q\rangle \Downarrow q[x := n]}$$

$$\frac{\langle b, q\rangle \Downarrow \text{true} \qquad \langle s_1, q\rangle \Downarrow q'}{\langle \textbf{if } b \textbf{ then } s_1 \textbf{ else } s_2, q\rangle \Downarrow q'}$$

$$\frac{\langle b, q\rangle \Downarrow \text{false} \qquad \langle s_2, q\rangle \Downarrow q'}{\langle \textbf{if } b \textbf{ then } s_1 \textbf{ else } s_2, q\rangle \Downarrow q'}$$

$$\frac{\langle b, q\rangle \Downarrow \text{false}}{\langle \textbf{while } b \textbf{ do } s, q\rangle \Downarrow q}$$

$$\frac{}{\langle \textbf{havoc } x, q\rangle \Downarrow q[x := n]}$$

$$\frac{\langle b, q\rangle \Downarrow \text{true} \qquad \langle s \,;\, \textbf{while } b \textbf{ do } s, q\rangle \Downarrow q'}{\langle \textbf{while } b \textbf{ do } s, q\rangle \Downarrow q'}$$

UNIVERSITY OF WATERLOO

Symbolic States and Path Conditions

- A *symbolic state* (or symbolic environment) q is a map from program variables to *symbolic expressions*
- A *path condition* is a formula over symbolic expressions
- A *judgment* in symbolic execution has a form

$$\langle s, q, pc \rangle \Downarrow q', pc'$$

where s is a statement, q and q' are the input and output symbolic environments, respectively, and pc and pc' are input and output path conditions, respectively

A path condition pc' is satisfiable iff there is a concrete input state c and concrete output state c' such that s started in state c reaches state c', i.e., $\langle s, c \rangle \Downarrow c'$

Symbolic Semantics: Expressions

$$\frac{}{\langle n, q\rangle \Downarrow n}\, n \in \mathbb{Z} \qquad \frac{}{\langle v, q\rangle \Downarrow q(v)}\, v \in L$$

$$\frac{\langle a_1, q\rangle \Downarrow n_1 \qquad \langle a_2, q\rangle \Downarrow n_2}{\langle a_1 + a_2, q\rangle \Downarrow \mathsf{Plus}(n_1, n_2)} \qquad \frac{\langle a_1, q\rangle \Downarrow n_1 \qquad \langle a_2, q\rangle \Downarrow n_2}{\langle a_1 \times a_2, q\rangle \Downarrow \mathsf{Times}(n_1, n_2)}$$

$$\frac{\langle a_1, q\rangle \Downarrow n_1 \qquad \langle a_2, q\rangle \Downarrow n_2}{\langle a_1 < a_2, q\rangle \Downarrow \mathsf{Lt}(n_1, n_2)} \qquad \frac{\langle a_1, q\rangle \Downarrow n_1 \qquad \langle a_2, q\rangle \Downarrow n_2}{\langle a_1 > a_2, q\rangle \Downarrow \mathsf{Gt}(n_1, n_2)}$$

$$\frac{\langle b_1, q\rangle \Downarrow r_1 \qquad \langle b_2, q\rangle \Downarrow r_2}{\langle b_1 \wedge b_2, q\rangle \Downarrow \mathsf{And}(r_1, r_2)} \qquad \frac{\langle b_1, q\rangle \Downarrow r_1 \qquad \langle b_2, q\rangle \Downarrow r_2}{\langle b_1 \vee b_2, q\rangle \Downarrow \mathsf{Or}(r_1, r_2)}$$

UNIVERSITY OF WATERLOO

Symbolic Semantics: Statements (1/2)

$$\frac{}{\langle \textbf{skip}, q, pc \rangle \Downarrow q, pc}$$

$$\frac{}{\langle \textbf{print\_state}, q, pc \rangle \Downarrow q, pc}$$

$$\frac{\langle s_1, q, pc \rangle \Downarrow q'', pc'' \qquad \langle s_2, q'', pc'' \rangle \Downarrow q', pc'}{\langle s_1 \,;\, s_2, q, pc \rangle \Downarrow q', pc'}$$

$$\frac{\langle e, q \rangle \Downarrow r}{\langle x := e, q, pc \rangle \Downarrow q[x := r], pc}$$

$$\frac{R \text{ is a fresh symbolic constant}}{\langle \textbf{havoc } x, q, pc \rangle \Downarrow q[x := R], pc}$$

Symbolic Semantics: Statements (2/2)

$$\frac{\langle b, q\rangle \Downarrow r \qquad pc \wedge r \text{ is SAT} \qquad \langle s_1, q, pc \wedge r\rangle \Downarrow q', pc'}{\langle \textbf{if } b \textbf{ then } s_1 \textbf{ else } s_2, q, pc\rangle \Downarrow q', pc'}$$

$$\frac{\langle b, q\rangle \Downarrow r \qquad pc \wedge \neg r \text{ is SAT} \qquad \langle s_2, q, pc \wedge \neg r\rangle \Downarrow q', pc'}{\langle \textbf{if } b \textbf{ then } s_1 \textbf{ else } s_2, q, pc\rangle \Downarrow q', pc'}$$

$$\frac{\langle b, q\rangle \Downarrow r \qquad pc \wedge \neg r \text{ is SAT}}{\langle \textbf{while } b \textbf{ do } s, q, pc\rangle \Downarrow q, pc \wedge \neg r}$$

$$\frac{pc \wedge r \text{ is SAT} \qquad \dfrac{\langle b, q\rangle \Downarrow r \qquad \langle s\,;\textbf{while } b \textbf{ do } s, q, pc \wedge r\rangle \Downarrow q', pc'}{\langle \textbf{while } b \textbf{ do } s, q, pc\rangle \Downarrow q', pc'}}{\langle \textbf{while } b \textbf{ do } s, q, pc\rangle \Downarrow q', pc'}$$

Concolic Semantics: Notation

- Program variables L are partitioned into symbolic and concrete

$$Sym(L) = \text{symbolic} \qquad Con(L) = \text{concrete}$$

- Possibly there are no symbolic (or concrete) variables

- $Sym(a)$ and $Con(a)$ indicates whether a given variable a is symbolic or concrete, respectively.

UNIVERSITY OF WATERLOO

Concolic States

- A concolic state is a triple

$$q = \langle c, s, pc \rangle$$

where c is concrete, s is symbolic, and pc is a path condition

$$con(q) = c \qquad sym(q) = s \qquad pc(q) = pc$$

- every variable v in $sym(q)$ has a *concrete shadow* $con(q)(v)$. That is, concrete state can evaluate all variables

UNIVERSITY OF WATERLOO

Equivalence and containment

Equivalence of concrete states

Two concrete states c_1 and c_2 are equivalent, $c_1 \equiv_{con} c_2$, whenever they agree on concrete variables:

$$c_1 \equiv_{con} c_2 \iff \forall a \in Con(L) \cdot c_1(a) = c_2(a)$$

Containment of concrete states

$c \models \langle s, pc \rangle$ means that concrete c is contained in symbolic $\langle s, pc \rangle$

UNIVERSITY OF WATERLOO

Concolic Semantics of Expressions

- The semantics of expressions is as usual with variables evaluated based on their kind: concrete variables are evaluated over $con(q)$ and symbolic over $sym(q)$:

$$\frac{con(a) \qquad \langle a, con(q)\rangle \Downarrow v}{\langle a, q\rangle \Downarrow v} \qquad \frac{sym(a) \qquad \langle a, sym(q)\rangle \Downarrow v}{\langle a, q\rangle \Downarrow v}$$

- Expressions that do not depend on symbolic variables are evaluated concretely

- Expressions that depend on symbolic variables are evaluated symbolically (i.e., their value is some AST)

Concolic Semantics: Simple Statements

For most statements, the semantics is extended by applying both symbolic and concrete operational semantics in parallel:

$$\frac{\langle s, con(q)\rangle \Downarrow c \qquad \langle s, sym(q), pc(q)\rangle \Downarrow s', pc' \qquad c \models \langle s', pc'\rangle}{\langle s, q\rangle \Downarrow \langle c, s', pc'\rangle}$$

The last pre-condition ensures that the concrete and symbolic states are chosen consistently.

UNIVERSITY OF WATERLOO

Concolic Semantics: Assignment

Assignment of values to concrete variables is limited to concrete values only:

$$\frac{\langle e, q\rangle \Downarrow n \qquad con(x) \qquad n \in \mathbb{Z}}{\langle x := e, q\rangle \Downarrow q[x := n]}$$

- It is not possible to assign symbolic variables (or symbolic expressions) to concrete variables!
- To assign a symbolic value to a concrete variable either
  - make concrete variable symbolic (always easy to do), or
  - make symbolic value concrete (described later on)
- Assignment to symbolic variables also assigns to their concrete shadows.

Concolic Semantics: If-statement

At if-statement, concolic execution can chose to switch to the branch that is not consistent with current concrete state, as long as the concrete state can be adjusted. We only show one of the cases:

$$\frac{\begin{array}{ccc} & \langle b, con(q)\rangle \Downarrow \text{true} & \\ \langle b, q\rangle \Downarrow r \qquad pc(q) \wedge \neg r \text{ is SAT} & & c \models \langle sym(q), pc(q) \wedge \neg r\rangle \\ c \equiv_{con} con(q) & \langle s_2, \langle c, sym(q), pc(q) \wedge \neg r\rangle\rangle \Downarrow q' & \end{array}}{\langle \textbf{if } b \textbf{ then } s_1 \textbf{ else } s_2, q\rangle \Downarrow q'}$$

Concolic Semantics: Concretization

Concretization turns symbolic variables (or values) into the values of their concrete shadows. The choice of concretization is captured in a *concretization constraints* in the path condition:

$$\frac{sym(x) \qquad \langle x, con(q)\rangle \Downarrow n \qquad \langle x, sym(q)\rangle \Downarrow r \qquad \langle s, \langle con(q), sym(q)[x := n], pc(q) \wedge r = n\rangle\rangle \Downarrow q'}{\langle s, q\rangle \Downarrow q'}$$

That is, if x is a symbolic variable with symbolic value r and it is currently shadowed concretely by a concrete value n, then its value can be concretized to n as long as the path condition is updated with $r = n$ to reflect the concretization