SYMBOLIC EXECUTION AND PROGRAM VERIFICATION

- We present symbolic execution in SOS as abstract interpretation in the syntactic domain of arithmetic and Boolean expressions, but now over logical variables.

- A symbolic configuration $\langle S, ss, pc \rangle$ consists of:
  a statement $S \in Stm$
  a symbolic state $ss : Var \to AExp_{LVar}$ initially $x \mapsto x_0$
  a path condition $pc \in BExp_{LVar}$ initially true

- We extend While with three statements:

  assume b : halts if b doesn't hold
  assert b : aborts if b doesn't hold
  havoc X : assigns random values to the variables in $X \subseteq Var$

- The intention is that

  $\models_{par} \{P\} S \{Q\}$ if assume P; S; assert Q has no aborting executions

- Complication: executions can be infinite. Possible ways of addressing while loops:
  - unfolding some fixed number k of times (unsound)
  - using user-provided loop invariants to convert into loop-free program

- A path is feasible if all involved path conditions are satisfiable

- Rules for symbolic execution

$[\mathrm{skip}_{SE}]$ $\quad \langle \mathrm{skip}, ss, pc\rangle \Rightarrow \langle ss, pc\rangle$

$[\mathrm{ass}_{SE}]$ $\quad \langle x := a, ss, pc\rangle \Rightarrow \langle ss[x \mapsto \mathcal{SA}[\![a]\!](ss)], pc\rangle$

$[\mathrm{asm}_{SE}]$ $\quad \langle \mathrm{assume}\ P, ss, pc\rangle \Rightarrow \langle ss, pc \wedge \mathcal{SB}[\![P]\!](ss)\rangle$

$[\mathrm{asr}_{SE}]$ $\quad \langle \mathrm{assert}\ P, ss, pc\rangle \Rightarrow \langle ss, pc\rangle$ if $pc \Rightarrow \mathcal{SB}[\![P]\!](ss)$

$[\mathrm{hav}_{SE}]$ $\quad \langle \mathrm{havoc}\ X, ss, pc\rangle \Rightarrow \langle ss[x^1 \mapsto x_0^1, \ldots, x^n \mapsto x_0^n], pc\rangle$

where $\{x^1, \ldots, x^n\} = X$ and $x_0^1, \ldots, x_0^n$ are fresh

$[\mathrm{comp}^1_{SE}]$
$$\frac{\langle S_1, ss, pc\rangle \Rightarrow \langle ss', pc'\rangle}{\langle S_1; S_2, ss, pc\rangle \Rightarrow \langle S_2, ss', pc'\rangle}$$

$[\mathrm{comp}^2_{SE}]$
$$\frac{\langle S_1, ss, pc\rangle \Rightarrow \langle S_1', ss', pc'\rangle}{\langle S_1; S_2, ss, pc\rangle \Rightarrow \langle S_1'; S_2, ss', pc'\rangle}$$

$[\mathrm{if}^1_{SE}]$ $\quad \langle \underline{\mathrm{if}}\ b\ \underline{\mathrm{then}}\ S_1\ \underline{\mathrm{else}}\ S_2, ss, pc\rangle \Rightarrow \langle S_1, ss, pc \wedge \mathcal{SB}[\![b]\!](ss)\rangle$

$[\mathrm{if}^2_{SE}]$ $\quad \langle \underline{\mathrm{if}}\ b\ \underline{\mathrm{then}}\ S_1\ \underline{\mathrm{else}}\ S_2, ss, pc\rangle \Rightarrow \langle S_2, ss, pc \wedge \mathcal{SB}[\![\neg b]\!](ss)\rangle$

$[\mathrm{while}^1_{SE}]$ $\quad \langle \underline{\mathrm{while}}\ b\ \underline{\mathrm{do}}\ S, ss, pc\rangle \Rightarrow \langle S; \underline{\mathrm{while}}\ b\ \underline{\mathrm{do}}\ S, ss, pc \wedge \mathcal{SB}[\![b]\!](ss)\rangle$

$[\mathrm{while}^2_{SE}]$ $\quad \langle \underline{\mathrm{while}}\ b\ \underline{\mathrm{do}}\ S, ss, pc\rangle \Rightarrow \langle ss, pc \wedge \mathcal{SB}[\![\neg b]\!](ss)\rangle$

where: $\quad \mathcal{SA}[\![a]\!](ss) \stackrel{def}{=} a[ss] \qquad \mathcal{SB}[\![b]\!](ss) \stackrel{def}{=} b[ss]$

EXERCISE 2 : Verify $\{x \leq 0\}$ if $0 \leq x$ then $x := x+1$ else $x := 1$ $\{x=1\}$

Becomes:

$$\left.\text{assume } x \leq 0;\right\} \quad M$$

$$\left.\begin{array}{l} \underline{\text{if}}\ 0 \leq x\ \underline{\text{then}} \\ \quad x := x+1 \\ \underline{\text{else}} \\ \quad x := 1; \end{array}\right\} \quad I$$

$$\left.\text{assert } x = 1\right\} \quad R$$

$$\langle M; I; R,\ ss[x \mapsto x_0],\ true \rangle$$

$$\Downarrow$$

$$\langle I; R,\ ss[x \mapsto x_0],\ x_0 \leq 0 \rangle$$

$$\swarrow \qquad\qquad \searrow$$

$$\langle x := x+1; R,\ ss[x \mapsto x_0],\ x_0 \leq 0 \wedge 0 \leq x_0 \rangle \qquad \langle x := 1; R,\ ss[x \mapsto x_0],\ x_0 \leq 0 \wedge \neg(0 \leq x_0) \rangle$$

$$\Downarrow \qquad\qquad \Downarrow$$

$$\langle R,\ ss[x \mapsto x_0+1],\ x_0 \leq 0 \wedge 0 \leq x_0 \rangle \qquad \langle R,\ ss[x \mapsto 1],\ x_0 \leq 0 \wedge \neg(0 \leq x_0) \rangle$$

$$\Downarrow ① \qquad\qquad \Downarrow ②$$

$$\langle ss[x \mapsto x_0+1],\ x_0 \leq 0 \wedge 0 \leq x_0 \rangle \qquad \langle ss[x \mapsto 1],\ x_0 \leq 0 \wedge \neg(0 \leq x_0) \rangle$$

① $x_0 \leq 0 \wedge 0 \leq x_0 \Rightarrow x_0 + 1 = 1$ ② $x_0 \leq 0 \wedge \neg(0 \leq x_0) \Rightarrow 1 = 1$

Both paths are feasible, since final path conditions are satisfiable. And none of the (feasible) paths aborts, since both verification conditions are valid. Therefore the program is correct w.r.t. its specification.

Note that we can annotate programs at any control point! This makes a connection to the tableax (proof outlines) which we considered as fully annotated programs.

EXAMPLE 1

Program SWAP $\stackrel{def}{=}$ $x := x+y;\ y := x-y;\ x := x-y$

Spec $\{true\}$ SWAP $\{x = y_0 \land y = x_0\}$

Converted program, with labels:

l_0 : assume true;
l_1 : $x := x+y$;
l_2 : $y := x-y$;
l_3 : $x := x-y$;
l_4 : assert $x = y_0 \land y = x_0$;
l_F :

Symbolic execution:

$$\begin{aligned}
&\langle l_0, [x \mapsto x_0, y \mapsto y_0], true\rangle \\
\Rightarrow\ &\langle l_1, [x \mapsto x_0, y \mapsto y_0], true\rangle \\
\Rightarrow\ &\langle l_2, [x \mapsto x_0+y_0, y \mapsto y_0], true\rangle \\
\Rightarrow\ &\langle l_3, [x \mapsto x_0+y_0, y \mapsto (x_0+y_0)-y_0], true\rangle \\
\Rightarrow\ &\langle l_4, [x \mapsto (x_0+y_0)-((x_0+y_0)-y_0), y \mapsto (x_0+y_0)-y_0], true\rangle \\
\stackrel{(1)}{\Rightarrow}\ &\langle l_F, [\quad - \ " \ - \quad], true\rangle
\end{aligned}$$

(1) $\quad true \Rightarrow (x_0+y_0)-((x_0+y_0)-y_0) = y_0 \land (x_0+y_0)-y_0 = x_0$ ✓

EXAMPLE 2

Verify $\{x \le 0\}$ if $0 \le x$ then $x := x+1$ else $x := 1$ $\{x = 1\}$

- Loop elimination

A while loop annotated with a candidate invariant

$$\{I\}\ \underline{\text{while}}\ b\ \underline{\text{do}}\ S$$

is translated to:

```
assert I;
havoc X_s;
assume I;
if b then
    S;
    assert I;
    assume false;
else
    skip;
```

where X_s is the set of all variables that are assigned to in S

EXAMPLE 3

Program Pow $\stackrel{def}{=}$ $p := 1$; while $\neg(y=0)$ do $p := p * x$; $y := y - 1$

Spec $\{true\}$ Pow $\{p = x_0^{y_0}\}$

Invariant $p * x^y = x_0^{y_0}$

Converted program, with labels:

l_0: assume true;

l_1: $p := 1$;

l_2: assert $p * x^y = x_0^{y_0}$;

l_3: havoc p, y;

l_4: assume $p * x^y = x_0^{y_0}$;

l_5: if $\neg(y=0)$ then

l_6: $p := p * x$;

l_7: $y := y - 1$;

l_8: assert $p * x^y = x_0^{y_0}$;

l_9: assume false;

else

l_{10}: skip

l_{11}: assert $p = x_0^{y_0}$

l_F:

Symbolic execution:

$$
\begin{aligned}
&\langle l_0, [x \mapsto x_0, y \mapsto y_0, p \mapsto p_0], true \rangle \\
\Rightarrow\ &\langle l_1, [x \mapsto x_0, y \mapsto y_0, p \mapsto p_0], true \rangle \\
\Rightarrow\ &\langle l_2, [x \mapsto x_0, y \mapsto y_0, p \mapsto 1], true \rangle \\
\overset{(1)}{\Rightarrow}\ &\langle l_3, [x \mapsto x_0, y \mapsto y_0, p \mapsto 1], true \rangle \\
\Rightarrow\ &\langle l_4, [x \mapsto x_0, y \mapsto y_0', p \mapsto p_0'], true \rangle \\
\Rightarrow\ &\langle l_5, [x \mapsto x_0, y \mapsto y_0', p \mapsto p_0'], p_0' * x_0^{y_0'} = x_0^{y_0} \rangle
\end{aligned}
$$

$$
\begin{aligned}
\Rightarrow\ &\langle l_6, [x \mapsto x_0, y \mapsto y_0', p \mapsto p_0'], p_0' * x_0^{y_0'} = x_0^{y_0} \wedge \neg(y_0' = 0) \rangle \\
\Rightarrow\ &\langle l_7, [x \mapsto x_0, y \mapsto y_0', p \mapsto p_0' * x_0], \quad \text{---''---} \rangle \\
\Rightarrow\ &\langle l_8, [x \mapsto x_0, y \mapsto y_0' - 1, p \mapsto p_0' * x_0], \quad \text{---''---} \rangle \\
\overset{(2)}{\Rightarrow}\ &\langle l_9, [\quad \text{---''---} \quad], \quad \text{---''---} \rangle \\
\Rightarrow\ &\langle l_{11}, [\quad \text{---''---} \quad], \quad \text{---''---} \wedge false \rangle
\end{aligned}
$$

infeasible!

$$
\begin{aligned}
\Rightarrow\ &\langle l_{10}, [x \mapsto x_0, y \mapsto y_0', p \mapsto p_0'], p_0' * x_0^{y_0'} = x_0^{y_0} \wedge \neg\neg(y_0' = 0) \rangle \\
\Rightarrow\ &\langle l_{11}, [\quad \text{---''---} \quad], \quad \text{---''---} \rangle \\
\overset{(3)}{\Rightarrow}\ &\langle l_F, [\quad \text{---''---} \quad], \quad \text{---''---} \rangle
\end{aligned}
$$

(1) $\quad true \Rightarrow 1 * x_0^{y_0} = x_0^{y_0} \quad \checkmark$

(2) $\quad p_0' * x_0^{y_0'} = x_0^{y_0} \wedge \neg(y_0' = 0) \Rightarrow (p_0' * x_0) * x_0^{(y_0' - 1)} = x_0^{y_0} \quad \checkmark$

(3) $\quad p_0' * x_0^{y_0'} = x_0^{y_0} \wedge \neg\neg(y_0' = 0) \Rightarrow p_0' = x_0^{y_0} \quad \checkmark$