SYMBOLIC EXECUTION

- Symbolic Execution can be seen as Abstract Interpretation in the symbolic domain of expressions. E.g., x plus 1 gives as result the syntactic term $x+1$. We add to configurations a third component called a path condition, which collects the symbolic Boolean guards along executions.

Example:

$$\Downarrow \qquad \text{(logical variable} \in \text{LVar)}$$

$$\langle x := x+1 \,;\, \underline{\text{if}}\ 0 \le x\ \underline{\text{then}}\ x := x-1\ \underline{\text{else}}\ \text{skip}\,,\ [x \mapsto x_0],\ \text{true}\rangle$$

$$\Downarrow$$

$$\langle \underline{\text{if}}\ 0 \le x\ \underline{\text{then}}\ x := x-1\ \underline{\text{else}}\ \text{skip}\,,\ [x \mapsto x_0+1],\ \text{true}\rangle$$

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

$$\langle x := x-1\,,\ [x \mapsto x_0+1],\ 0 \le x_0+1\rangle \qquad \langle \text{skip}\,,\ [x \mapsto x_0+1],\ \neg(0 \le x_0+1)\rangle$$

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

$$\langle [x \mapsto (x_0+1)-1],\ 0 \le x_0+1\rangle \qquad \langle [x \mapsto x_0+1],\ \neg(0 \le x_0+1)\rangle$$

We obtained two paths. A path is called <u>feasible</u> if its path condition is a satisfiable Boolean formula.

- Testing: Symbolic execution was developed originally, in 1976 by James King (see Course Literature) for testing purposes:
  - pick a (path) coverage criterion CC;
  - for the given program S, generate all paths Paths^S_{CC} in S according to criterion CC; (finitely many, of finite length)
  - symbolically execute all paths in Paths^S_{CC};
  - collect the corresponding final path conditions PConds^S_{CC};
  - for each $pc \in \text{PConds}^S_{CC}$, use SAT to find a satisfying assignment

• The Formal Set-up

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

where $AExp_{SE}$ and $BExp_{SE}$ are like $AExp$ and $BExp$, but over logical variables and (semantic) integers and truth values.

LVar $\quad\quad$ $\mathbb{Z}$ $\quad\quad$ $\mathbb{T}$

Evaluation $\mathcal{SA}[\![a]\!](ss)$ is simply $a[ss]$, i.e., the arithmetic expression a where the variables are substituted with expressions according to ss. For example $\mathcal{SA}[\![x-1]\!]([x \mapsto x_0+1]) = (x_0+1)-1$.
Similarly, $\mathcal{SB}[\![b]\!](ss) \stackrel{def}{=} b[ss]$.

We give an SOS-style symbolic execution (see rules).

- Later, we will extend While with additional statements and see how symbolic execution can be used for program verification.

- Rules for symbolic execution

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

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

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

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

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

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

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

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

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

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

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

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

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