Symbolic Execution

The verification method that VeriFast uses is called symbolic execution and is conceptually related to verification condition generation, but accomplishes the task by a different means.

Assert and Assume

First, we extend our programming language with three new statements:

assert(ϕ) – aborts if ϕ evaluates to false
assume(ϕ) – halts if ϕ evaluates to false (ignore such paths)
havoc(V) – assign arbitrary values to all variables $v \in V$

Then: $\vDash_{par} (\!|\phi|\!)\ C\ (\!|\psi|\!)$ if and only if the program assume(ϕ); C; assert(ψ) does not abort

Symbolic Execution

- we execute programs starting from a symbolic state, with symbolic constants as initial values (we use X, Y, ...)
- we consider all (feasible) paths from entry to exit point
- as we go along paths, we:
  - update symbolic state after assignment and havoc statements
  - update path condition when branching on Boolean guards and assumes:
  - substitute current symbolic values for the variables in the guard and add resulting formula to current path condition (check feasibility)
  - generate and prove proof obligations at assert statements; abort execution if false. Proof obligation is:
    $\vdash$ current_path_cond $\rightarrow$ assert_cond[symb_vals/vars]

We illustrate the technique on Abs:

$$(\!| x = x_0 |\!)\ \mathrm{if}\ (x > 0)\ \{y = x;\}\ \mathrm{else}\ \{y = -x;\}\ (\!| y = |x_0| |\!)$$

First, we transform the Hoare triple to a program. We introduce labels at control points to help following the paths:

```
L1: assume(x = x0);
L2: if(x > 0) {
L3:     y = x;
    } else {
L4:     y = -x;
    }
L5: assert(y = |x0|);
L6: return
```

We have two paths to consider. The first one is:

LABEL	SYMB. STATE	PATH COND.	PROOF OBLIG.
L_1	$x \mapsto X, y \mapsto Y$	true	
L_2	$x \mapsto X, y \mapsto Y$	$X = x_0$	
L_3	$x \mapsto X, y \mapsto Y$	$X = x_0 \wedge X > 0$	
L_5	$x \mapsto X, y \mapsto X$	$X = x_0 \wedge X > 0$	$\vdash X = x_0 \wedge X > 0 \rightarrow X = \lvert x_0 \rvert$
L_6	$x \mapsto X, y \mapsto X$	$X = x_0 \wedge X > 0$	

Compare the resulting proof obligation with the one from our proof in Hoare logic: $\vdash x = x_0 \wedge x > 0 \rightarrow x = |x_0|$

The second path $L_1 L_2 L_4 L_5 L_6$ is handled similarly: Exercise

Loops while $B\{C\}$ annotated with (candidate) invariants η are translated / re-written to the command sequence:

```
assert(η);
havoc(Var_C);
assume(η);
if B {
    C;
    assert(η);
    assume(false);
}else{
    skip;
}
```

where Var_C denotes all variables that are assigned to in the loop body C. Symbolic execution of havoc (Var_C) assigns new/fresh symbolic constants to the variables in Var_C. Then, the path condition can be simplified. The assume (false) statement above has the effect that the current path is not explored any further.

EXERCISE: Verify Copy, annotated with a loop invariant:

$(\!|x=x_0 \land x \geq 0|\!)\ y=0;\ (\!|x+y=x_0|\!)\ \text{while}\ (x!=0)\ \{y=y+1; x=x-1\}\ (\!|y=x_0|\!)$

VeriFast halts at points where
- current path condition is false : go to next path
- proof obligation is false : report an error

The presented method is equivalent to that of VCG, due to a fundamental equivalence for the assignment statement:

assertion $\psi[E/x]$ holds in a state if and only if ψ holds in the state resulting from s after executing the assignment $x = E$