```
def count_heads(prob : float, n : int):
        x = 0
        while(n > 0):
                d = bernoulli.rvs(size=1, p=prob)[0]
                x = x + d
                n = n - 1
        return x
```

Fig. 1. Program to *count* heads over n bernoulli trials.

```
if (prob <= 0.5) E(x) < 0.4 * n else E(x) >= 0.6 * n
```

Fig. 2. `Assert` we want to check for failure.

```
double prob, path_prob = 1, choice_prob = 1;
int sum = 0, n = 0;

klee_make_symbolic(&prob, f"prob_sym_{i}");
klee_make_symbolic(&n, f"n_symbolic");
for i in range(n):
        int d;
        klee_make_symbolic(&d, f"d_sym_{i}");

        d = bernoulli(prob);
        (d == 1) ? choice_prob = prob : choice_prob = (1 - prob);
        path_prob = path_prob * choice_prob;

        sum = sum + d;

klee_dump(path_prob)
klee_dump(sum) // E[heads in "n" runs]
```

Fig. 3. Program listing for n bernoulli trails experiment with transformation for KLEE.

We consider a independent *bernoulli* trials here of flipping a fair coin "n" times.

$$choice\_prob = \begin{cases} p & \text{if } d \text{ value is 1 corresponding to getting a "heads"} \\ 1-p & \text{if } d \text{ value is 0 corresponding to getting a "tails"} \end{cases}$$

concretely, on the i^{th} run $\vec{d}$ can have a value as below, one-hot encoded $w.r.t$ the outcome of *heads* or *tails*.

$$\vec{d_i} = encode(< 0, 0, 0, 1, 1, 0, 1, 1, 1, 0 >)$$

Based on the value of the $\vec{d_i}$, we get w_i value using *choice_prob*.

$$w_i = (p)^{x_i} * (1-p)^{n-x_i} \tag{1}$$

where x_i denotes the number of *heads* in the i^{th} randomized run and for n runs. `n = 10` for the case in the above example.

$$Objective_1 = maximize(\sum_{i=1}^{k} (p)^{x_i} * (1-p)^{n-x_i}) \quad | \; \forall (i,j) \;\; [x_i \neq x_j] \tag{4}$$

Fig. 4. Optimization Expression for k randomized paths

$$w_i = (p)^{x_i} * (1-p)^{10-x_i} \tag{2}$$

We consider top "k" randomized runs now for the optimization query. The expression for `optimization` thus becomes

$$maximize(\sum_{i=1}^{k} w_i)$$

On substituting the value of w_i from (1).

$$maximize(\sum_{i=1}^{k} (p)^{x_i} * (1-p)^{n-x_i}) \tag{3}$$

After performing the optimization above, we get different values of $\vec{d_i}$. For i^{th} randomized run. we get a `single` one hot encoded $\vec{d}$ vector. We show below the encoding for a few i values.

$$\vec{d_1} = encode(< 0, 0, 0, 1, 1, 0, 1, 1, 1, 0 >)$$
$$\vec{d_3} = encode(< 0, 1, 0, 1, 1, 0, 0, 0, 1, 0 >)$$
$$\vec{d_4} = encode(< 1, 0, 1, 0, 1, 0, 1, 1, 0, 1 >)$$
$$....$$

We run the optimization by renaming the k `pse` variables set appropriately and then impose the *distinct* clause so that we don't run the optimization on the same *randomized* runs again.

$$\forall (i,j) \;\; [\vec{d_i} \neq \vec{d_j}] \tag{5}$$

We approximate the value of *expected* heads in the above program, with the following equations.

$$w_i = \prod_{j=1}^{n} choice\_prob_i(j), \quad sum_i = \sum_{j=1}^{n} components(\vec{d_i}), \tag{6}$$

$$EV(heads) = (\sum_{i=1}^{k} w_i * sum_i), \quad Error = n * prob - EV(heads) \tag{7}$$

where both w_i and sum_i can both be computed from the corresponding $\vec{d_i}$ expression we get from the *model* of the *optimization* query Eq 4

For $k = 5$ & $n = 10$ the two `constraint` sets and `optimization` expressions are as follows :

$$w_1 = \prod_{j=1}^{10} choice\_prob_1(j), \quad sum_1 = \sum_{j=1}^{10} components(\vec{d_1}), \tag{8}$$

$$w_2 = \prod_{j=1}^{10} choice\_prob_2(j), \quad sum_2 = \sum_{j=1}^{10} components(\vec{d_2}), \tag{9}$$

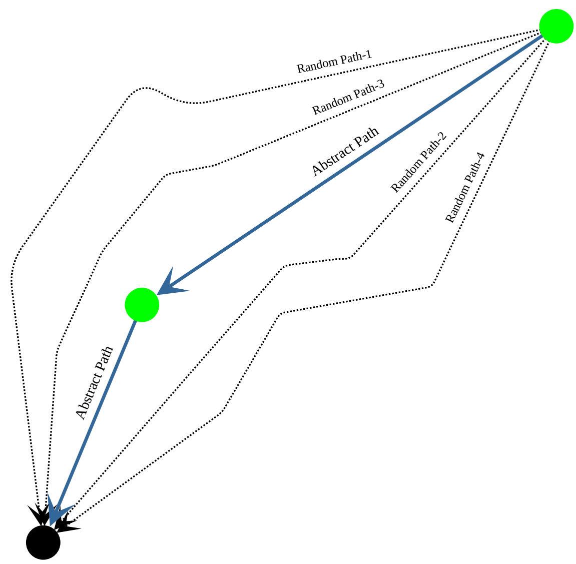

Fig. 5. Multiple *random* paths corresponding to different $\vec{d}$'s but a single *abstract* path.

```
prob = 0.435, n = 8 (8 coin flips), k = 240 (randomized paths)
prob = 0.551, n = 5 (5 coin flips), k = 27 (randomized paths)
```

Fig. 6. Values for `Assert` failure.

$$w_3 = \prod_{j=1}^{10} choice\_prob_3(j), \quad sum_3 = \sum_{j=1}^{10} components(\vec{d_3}), \tag{10}$$

$$w_4 = \prod_{j=1}^{10} choice\_prob_4(j), \quad sum_4 = \sum_{j=1}^{10} components(\vec{d_4}), \tag{11}$$

$$w_5 = \prod_{j=1}^{10} choice\_prob_5(j), \quad sum_5 = \sum_{j=1}^{10} components(\vec{d_5}), \tag{12}$$

$$\vec{d_1} \neq \vec{d_2} \neq \vec{d_3} \neq \vec{d_4} \neq \vec{d_5} \tag{13}$$

$$EV(heads) = (\sum_{i=1}^{k} w_i * sum_i), \tag{14}$$

$$EV(heads) = (w_1 * sum_1 \ + \ w_2 * sum_2 \ + \ w_3 * sum_3 \ + \ w_4 * sum_4 \ + \ w_5 * sum_5) \tag{15}$$

$$Objective_2 = maximize(w_1 * sum_1 \ + \ w_2 * sum_2 \ + \ w_3 * sum_3 \ + \ w_4 * sum_4 \ + \ w_5 * sum_5) \tag{16}$$

We can make multiple optimization *objectives* be fulfilled in one query, but the best results are produced for the $Objective_2$ since it minimizes the *error* most.

$$Objective_2 = maximize(\sum_{i=1}^{k} w_i * x_i) \tag{17}$$

1 QUICKSORT EXAMPLE

Consider a pivot vector $\vec{p_{abs}}$ that KLEE produces over a single abstract path. We consider a 10 element array as shown above. The *pivot* indices for $\vec{p_{abs}}$ as choosen from a *random* distribution.

$$arr[10] = \begin{array}{||c|c|c|c|c|c|c|c|c|c||} \hline a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9 \\ \hline \end{array} \tag{18}$$

A possible assignment of the indexes choosen as *pivot* indices along this *abstract* path can be as shown below.

$$\vec{p_{abs}} = \begin{array}{||c|c|c|c|c|c||} \hline a_9 & a_8 & a_2 & a_3 & a_8 & a_8 \\ \hline \end{array} \tag{19}$$

For same `forall` values in $arr[10]$, we can have different random runs and on each run, the $\vec{pivot}$ vector will have a different assignment of values as indices. We are showing 5 different $\vec{pvot}_{r_j}$ vectors, each corresponding to a valid random run j.

$$\vec{pvot}_{r_1} = \begin{array}{||c|c|c|c|c|c||} \hline a_9 & a_4 & a_4 & a_2 & a_9 & a_9 \\ \hline \end{array} \tag{20}$$

$$\vec{pvot}_{r_2} = \begin{array}{||c|c|c|c|c||} \hline a_9 & a_9 & a_7 & a_7 & a_7 \\ \hline \end{array} \tag{21}$$

$$\vec{pvot}_{r_3} = \begin{array}{||c|c|c|c|c|c||} \hline a_9 & a_4 & a_1 & a_4 & a_9 & a_9 \\ \hline \end{array} \tag{22}$$

$$\vec{pvot}_{r_4} = \begin{array}{||c|c|c|c|c|c||} \hline a_9 & a_3 & a_1 & a_9 & a_6 & a_9 \\ \hline \end{array} \tag{23}$$

$$\vec{pvot}_{r_5} = \begin{array}{||c|c|c|c|c|c||} \hline a_9 & a_5 & a_3 & a_1 & a_9 & a_8 \\ \hline \end{array} \tag{24}$$

$$\vec{pvot}_{r_6} = \begin{array}{||c|c|c|c|c|c|c||} \hline a_9 & a_2 & a_1 & a_9 & a_6 & a_6 & a_9 \\ \hline \end{array} \tag{25}$$

$$\vec{pvot}_{r_7} = \begin{array}{||c|c|c|c|c|c|c||} \hline a_9 & a_9 & a_9 & a_5 & a_5 & a_9 & a_8 \\ \hline \end{array} \tag{26}$$

The probability mass associated with the i_{th} random run as is given below considering each *pivot* selection being an independent selection. The variable $count_i$ denotes the number of comparisions made in t^{th} random run.

$$w_i = (1/n) * \prod_{j=1}^{k} 1/(\#size_{partition_j}) \quad | \quad for \;\; k - partitons \tag{27}$$

$$E[comparisions] = \sum_{i=1}^{r} (w_i * count_i) \quad | \quad for \;\; r - runs \tag{28}$$

2 PSE APPROACH

- Get the path contraints from KLEE, encoding each one into a SMT formula.
- A single *abstract* path may have multiple *randomized* runs, each corresponding to a different *setting* of the *probabilistic* variables.
- For each of the path $\in$ *abstract program paths*,
  - We treat the *probabilistic* variables as *nondet* vars.
  - Find the estimated Expected value of the *probabilistic* variable.
  - For each of the *probabilistic* run record the setting of for the *forall* variable.
  - We use these as *forall* inputs later for the *concrete* runs.
  - The constraints for each run is encoded into a *smt* formula. We make one call to Z3 at the end to get the value for all the randomized runs.
  - The Expected value will be an estimation here, we choose say the top k candidates only, reposnsible for *maximizing* $\sum_1^k w_i$ (Total Mass Probability).
- Now we do concrete runs of the program multiple times using the *forall* variable values from *randomized* runs.
- We use a *path* selection heuristic using a *maximizing* value of the forall setting that leads to a *violation.*