

Path-based SST relies on classical results from labelled combinatorial structures [11] to uniformly sample the set of program paths with length in  $[1, T]$ . Each path sample is provided to a constraint solver (oracle) and labelled as feasible or infeasible; see [9] and references therein. The infeasibility of a given path arises if it violates some dependencies between different parts of the program, referred to as *XOR patterns*. For instance if two `if` nodes are based on an unchanged expression, then their successors are correlated in every feasible path (if the program path includes the `then` successor of the first `if` node, it must also include the `then` successor of the second `if` node).

Because of the small number of available labelled paths (due to the labelling cost) compared to the complexity of the “natural” search space, i.e. that of long strings on a large alphabet, a frugal propositional representation inspired by Parikh maps [12] is considered. For  $t = 1 \dots T$ , let  $s[t]$  denote the  $t$ -th symbol in string  $s$ , set to value  $v_f$  if the length of  $s$  is less than  $T$ .

- To each symbol  $v$ , is associated an integer attribute  $a_v$ ;  $a_v(s)$  is the number of occurrences of symbol  $v$  in path  $s$ .
- To the  $i$ -th occurrence of a symbol  $v$ , is associated a categorical attribute  $a_{v,i}$ . Attribute  $a_{v,i}(s)$  gives the next ‘informative’ symbol following the  $i$ -th occurrence of symbol  $v$  in  $s$  (or  $v_f$  if  $s$  contains less than  $i$  occurrences of  $v$ ).

Preliminary attempts at discriminant learning have been hindered by the tiny percentage of the feasible paths, as could have been expected from [8]. A generative learning approach was then considered.

### 3 Overview of *EXIST*

This section describes a sampling algorithm called *EXIST* for *Exploration vs eXploitation Inference for Software Testing*, able to retrieve distinct feasible paths with high probability based on a set  $\mathcal{E}$  of feasible/infeasible paths.  $\mathcal{E}$ , initially set to a small set of labelled paths, is gradually enriched with the paths generated by *EXIST* and labelled by the constraint solver.

*EXIST* proceeds by iteratively exploiting and updating a probabilistic model  $\mathcal{P}$ . *EXIST* involves two modules: the *Init* module estimates the probability for a path to be feasible conditionally to its extended Parikh description<sup>2</sup>; the *Decision* module uses the  $\mathcal{P}$  model to iteratively construct the current path  $s$ .

#### 3.1 Decision module

Let  $s$  (resp.  $v$ ) denote the path under construction (resp. the last node symbol in  $s$ ). Let  $i$  be the total number of occurrences of  $v$  in  $s$ . Let  $w$  be one possible successor node of  $v$ ; if  $w$  is selected, the total number of  $w$  symbols in the final path will be at least the current number of occurrences of  $w$  in  $s$ , plus one; let  $\eta_w$  denote this number.

Let us define  $p_s(w)$  as the probability for a path  $S$  to be feasible conditionally to  $E_{s,w}(S) = [a_{v,i}(S) \equiv w] \wedge [a_w(S) \geq \eta_w]$ , estimated by the *Init* module;  $p_s(w)$  is conventionally set to 1 if there is no path in  $\mathcal{E}$  satisfying  $E_{s,w}$ .

Probabilities  $p_s(w)$  for  $w$  ranging over the successors of  $v$  are used to select the next node in  $s$ . Three options have been considered in order to favor the generation of a new feasible path.

The *Greedy* option selects the successor node  $w$  maximising  $p_s(w)$ .

The *RouletteWheel* option stochastically selects node  $w$  with probability proportional to  $p(s, w)$ .

The *BandiST* option considers the multi-armed bandit problem where every bandit arm corresponds to a successor  $w$  of the current node  $v$  and the associated reward is  $p_s(w)$ , and uses the UCB1 algorithm [1] for determining the best arm/successor node.

#### 3.2 Init module

The *Init* module determines how the conditional probabilities used by the *Decision* module are estimated. The baseline *Init* option computes  $p_s(w)$  as the fraction of paths in  $\mathcal{E}$  satisfying  $E_{s,w}$  that are feasible. However, this option fails to guide *EXIST* efficiently due to the disjunctive nature of the target concept, as shown on the following toy problem.

<sup>1</sup>Formally,  $a_{v,i}(s)$  is set to  $s[t(i) + k]$ , where  $t(i)$  is the index of the  $i$ -th occurrence of symbol  $v$  in  $s$ ;  $k$  is initially set to 1; in case  $a_{v,i}$  takes on a constant value over all examples,  $k$  is incremented.

<sup>2</sup>This probabilistic model space is meant to avoid the limitations of probabilistic FSAs and Variable Order Markov Models [4]. On one hand, probabilistic FSAs (and likewise simple Markov models) cannot model the long range dependencies of the *XOR patterns*. On the other hand, although Variable Order Markov Models can accommodate such dependencies, they are ill-suited to the sparsity of the initial data available.