Paper Review: Scaling Enumerative Program Synthesis via Divide and Conquer

Gaurav Mahajan

EUSolver uses formal specifications: SMT formulae (has to be plainly separable) as behavioral constraints and user defined conditional expression grammar as structural constraints. To search the space of candidate expressions, it enumerates the set of terms that covers all points and predicates, and builds a representation (decision tree using predicates as attributes and terms as labels) to find the expression which holds on all points.

Instead of looking through the entire search space, EUSolver decomposes the problem into finding smaller expressions that are correct on subsets of input and composes them with predicates (using decision trees). This speeds up the search primarily because of the observation that when expressions are large, the individual components are still small and can be quickly enumerated. To achieve this, EUSolver first finds a set of terms which covers all the points, then tries to learn a decision tree using the predicates as attributes and terms as labels. If such decision tree exists, then the corresponding expression holds on all points, else it adds more terms and predicates. Conditional expression grammars and plainly separability of the specification allows the decomposition onto a subset and the consequent composition. EUSolver also prunes the search tree by discarding terms that cover the same set of points, although this might lead to larger solutions.

The naive method for synthesizing branching would be to find a set of predicates S_t for each term t such that ($\forall$ pred $\in$ S_t pred holds on pt) $\iff$ pt $\in$ cover[t]. If such a set S_t exists $\forall$ t then the expression expr(T) corresponding to set of all terms T is defined as

- 0 if length(T) == 0
- (if (S_t) then t else expr(T-t)), otherwise, for any t in set T

where S_t holds on pt if pred holds on pt $\forall$ pred $\in$ S_t. By using Decision Tree with Information Gain heuristic, EUSolver is more probable to find expressions of smaller length and in less time.

References and Notes

1. Rajeev Alur, Arjun Radhakrishna, and Abhishek Udupa, *Scaling Enumerative Program Synthesis via Divide and Conquer.* Technical Report. University of Pennsylvania.