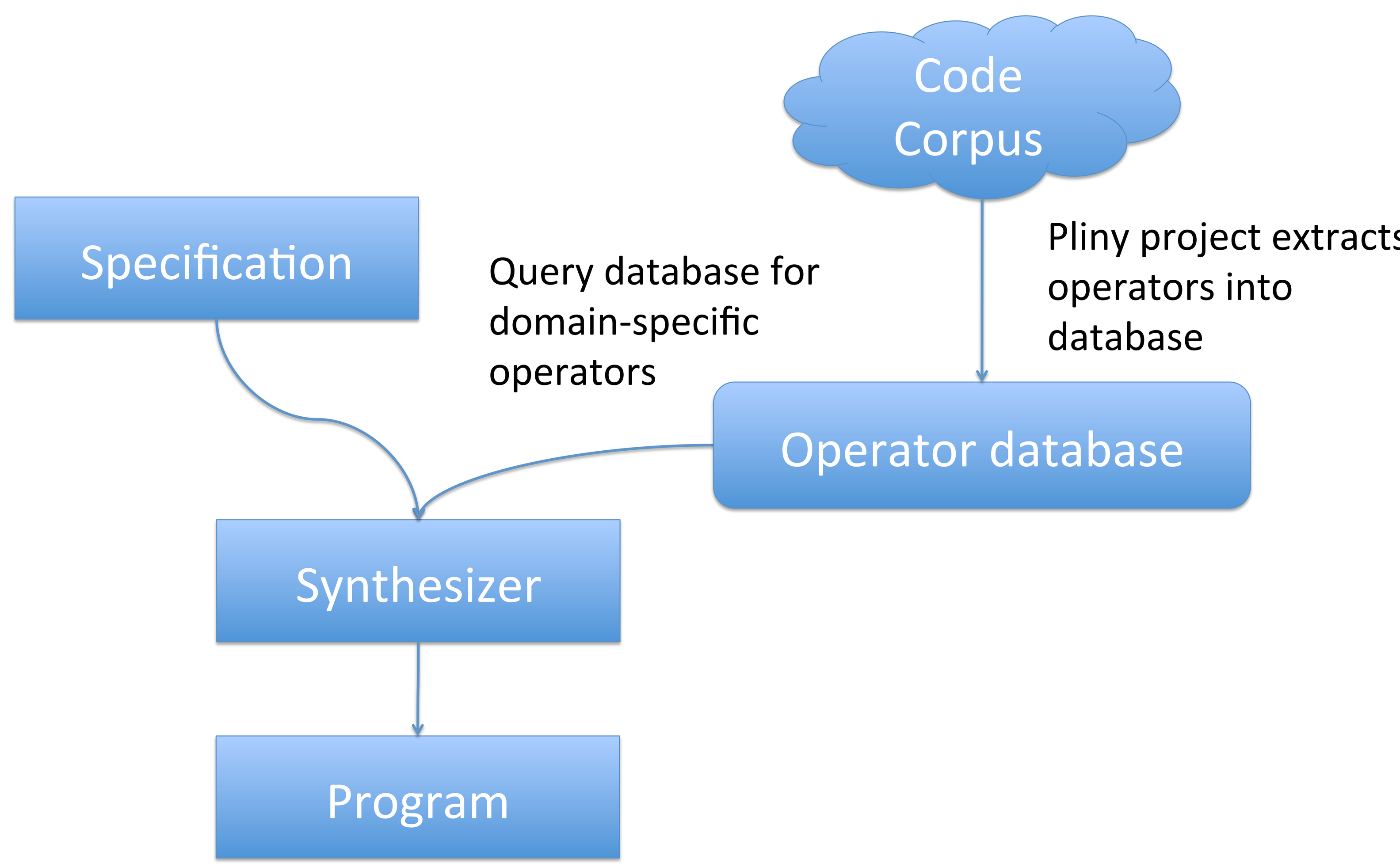
Unification and Partial Evaluation for Component based Synthesis

John Feser In collaboration with Vijayaraghavan Murali, Swarat Chaudhuri, and Isil Dillig

IDEA

Using operators from existing programs to perform synthesis means writing the synthesis algorithm only once and thus saves considerable time and resources.



Program specification

$$[\,] \to [\,]$$
$$[2] \to [2]$$
$$[2\ 1] \to [1\ 2]$$
$$[2\ 1\ 3] \to [3\ 1\ 2]$$

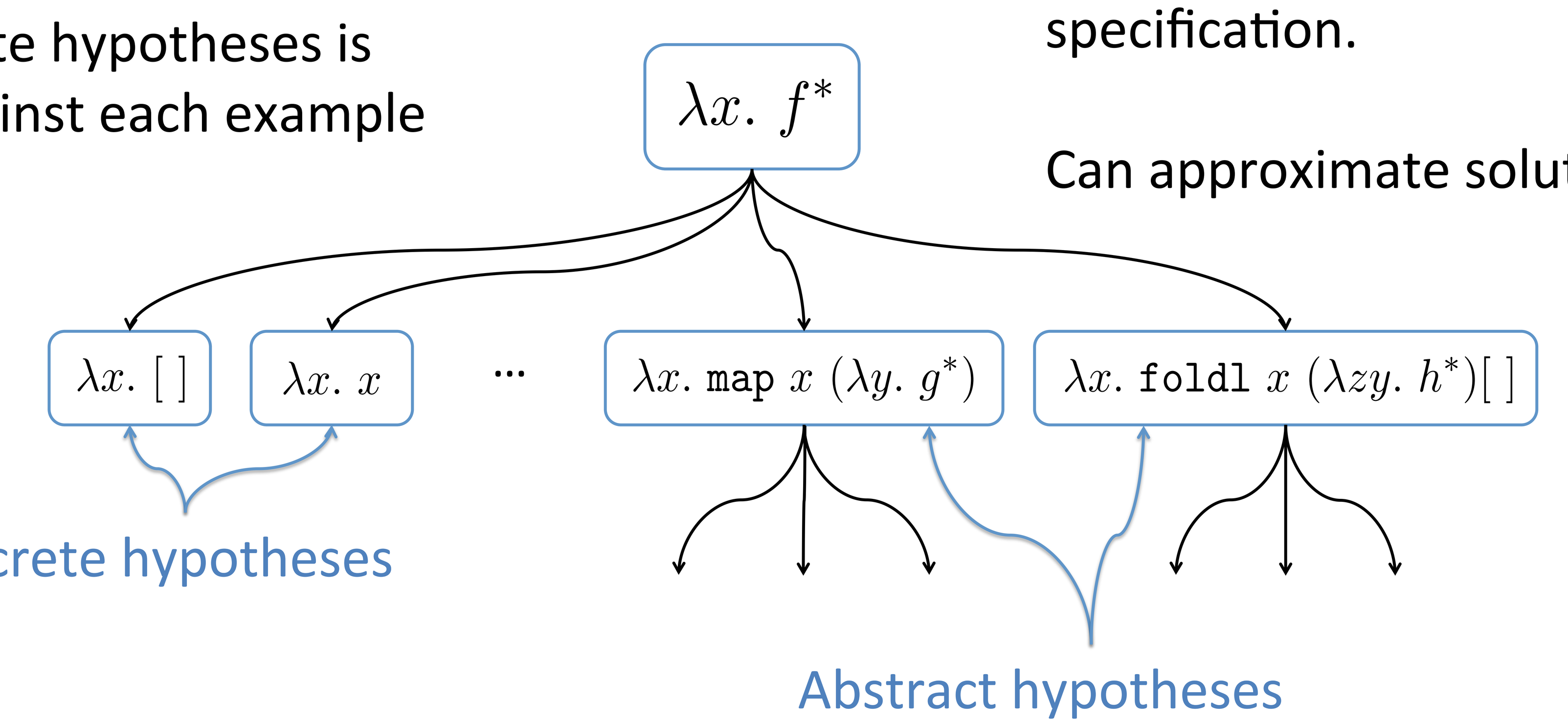
How can we prune an abstract hypothesis?

Must prove that no child can satisfy specification.

Can approximate solution using unification.

Pruning concrete hypotheses is easy: check against each example

Search tree



Child hypothesis ⟺ Assignment to holes in parent

PROBLEM

A large set of operators causes an explosion in the size of the search space.

SOLUTION

Use a new pruning technique to reduce the size of the search space, without relying on hand-coded information about operators.

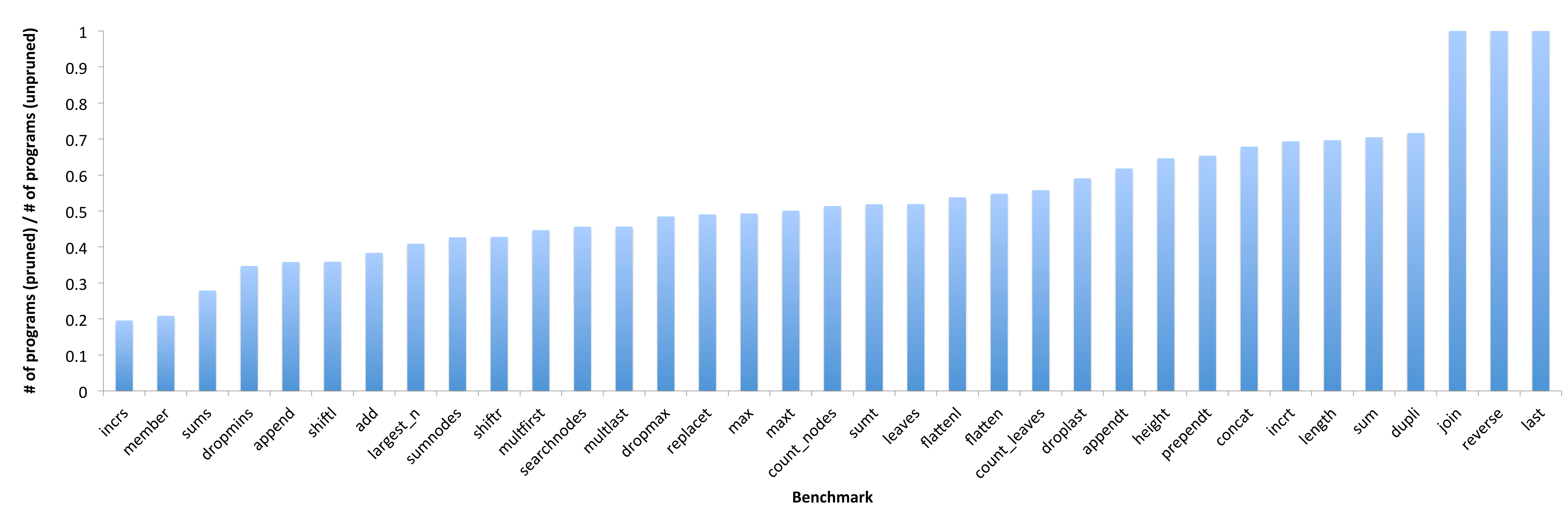
TECHNIQUE

For each abstract hypothesis H and input-output pair (i, o) determine whether there exists a substitution $\sigma = \{x_1 \leftarrow y_1 \dots x_n \leftarrow y_n\}$ such that $H[\sigma](i) = o$. In general, this is a higher-order unification problem, so is undecidable.

We can approximate the solution using partial evaluation followed by syntactic unification. For each (i, o), let H' be the result of partially evaluating $H(i)$. If H' does not unify with o, then σ does not exist. The partial evaluation step reduces the problem to first-order unification, which is efficiently decidable.

Fraction of programs remaining after pruning

46% average reduction in number of programs examined



14 June 2015