SynGuar: Guaranteeing Generalization in Programming by Example

Bo Wang, Teodora Baluta, Aashish Kolluri, Prateek Saxena
{bo_wang, teodora.baluta, aashishk}@u.nus.edu, prateeks@comp.nus.edu.sg

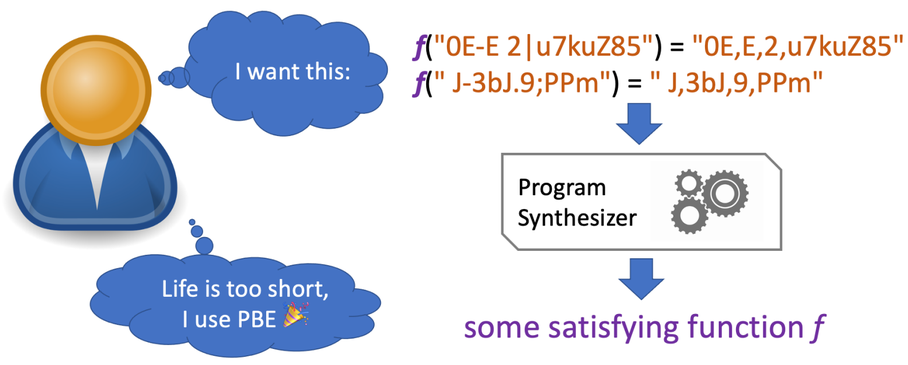
National University of Singapore

Abstract

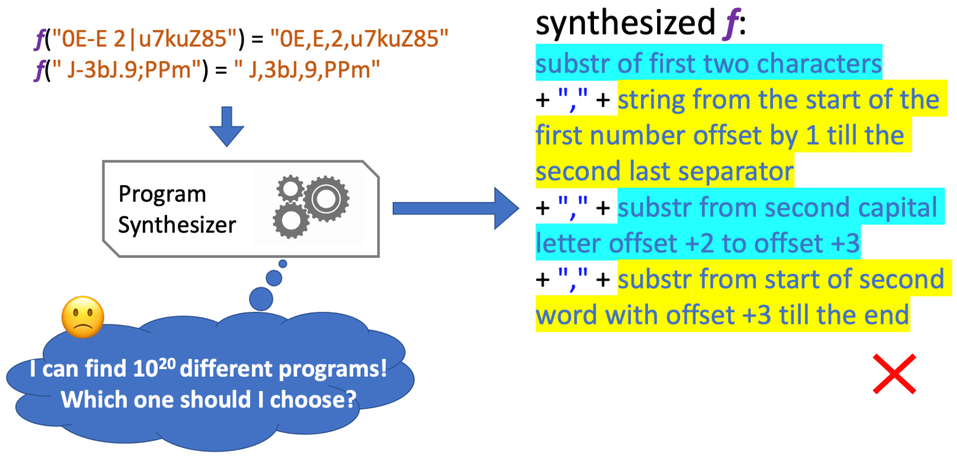
- **SynGuar** is a framework to provably reduce overfitting in program synthesis.
- It achieves **P**robability **A**pproximately **C**orrect (PAC) generalization in Programming-by-Example.
- It currently works for countable program space and realizable settings.

Overfitting Problem in Programing-by-Example

Programming-by-Example (PBE): Input-Output pairs ➔ Code



Synthesizers suffer from overfitting with **insufficient examples.**



PAC Framework

PAC framework gives a way to compute the **sufficient sample size for generalization.**

PAC Formalization

Synthesizer's hypothesis space H

How many examples are sufficient to return a program $f \in H$ that is **close to the target** with **high probability**?

The generalization error $error(f) = \Pr_{x\sim D}[f(x) \text{ not correct}]$ is bounded by ϵ.

The probability of generating f with $error(f) > \epsilon$ is bounded by δ.

Our Goal: Pr[*error(f)* > **ε**] < **δ**

Classical Sample Complexity Bound

with sample size $m > \frac{1}{\epsilon}(\ln|H| + \ln\frac{1}{\delta})$

we have Pr[*error(f)* > **ε**] < **δ**

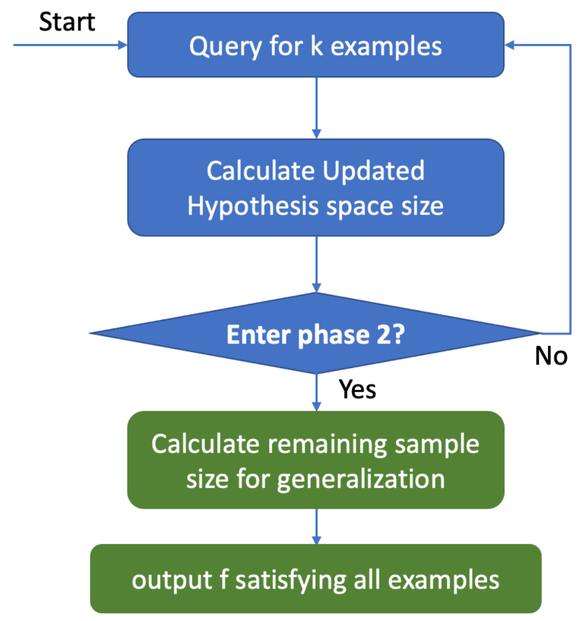
But synthesizers have **astronomical** program space sizes …

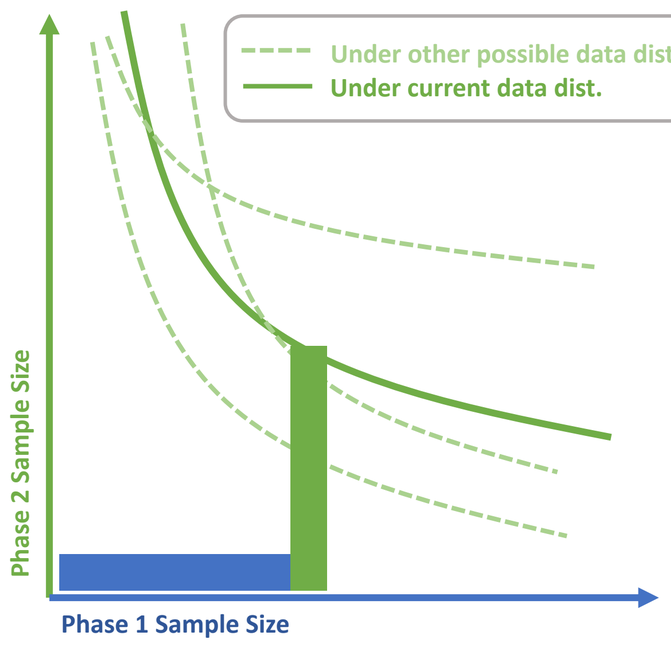
Program Synthesizer

```
language StrPROSE;
@input string x;
@start string program := recTerm;
string recTerm := catTerm | Concat(catTerm, recTerm);
string catTerm := ConstStr(cs) | convTerm;
string convTerm := term | UpperCase(term) | LowerCase(
     term);
string term := SubStr(x, pos1, pos2);
int? pos := AbsPos(x, ka) | RegPos(x, kr);
string cs; //constant string
int ka; //absolute position
Tuple<Regex,Regex,int,int> kr; // kr=(r1,r2,k,offset)
```

➡ $|H| > 10^{100}$

Solution: SynGuar Algorithm (Contribution 1)





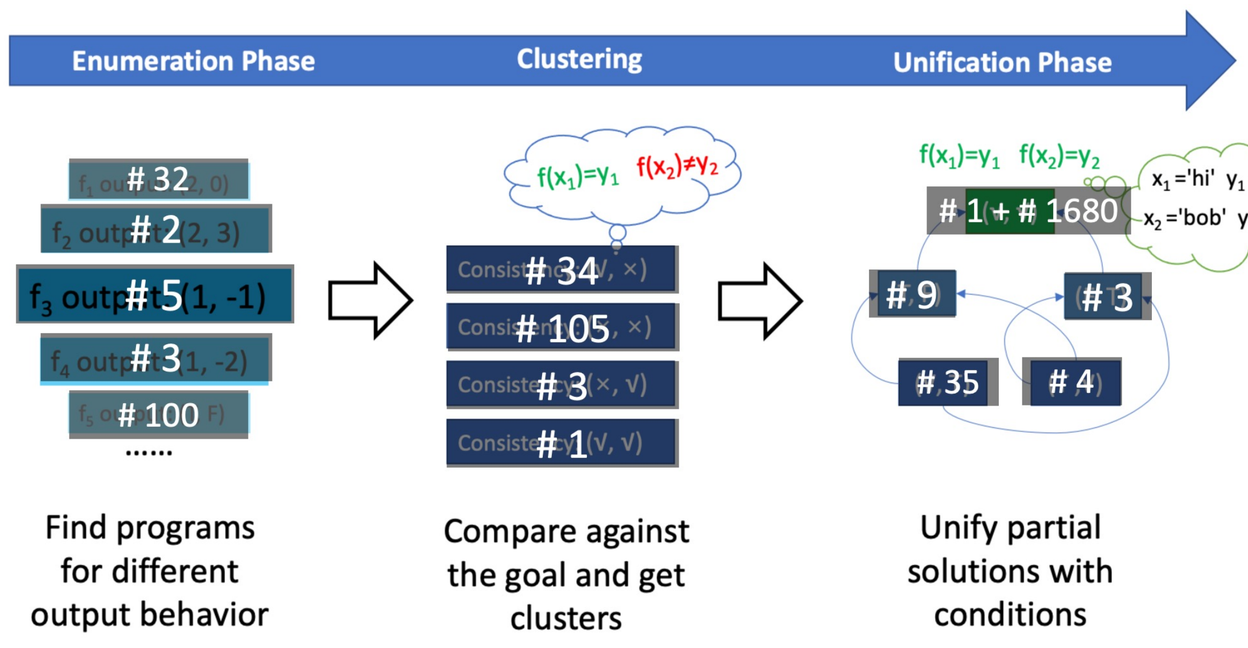
Goal: Minimize(▬ + ▮) **Achieved**: within 2x of the optimal

Challenge: Compute Program Space Size (Contribution 2)

- **PROSE-based** synthesizer (StrPROSE), already supported.

  PROSE framework has Size API for the program space.

- **Bottom-up** synthesizer (StrSTUN), modified to add counting.



Find programs for different output behavior

Compare against the goal and get clusters

Unify partial solutions with conditions

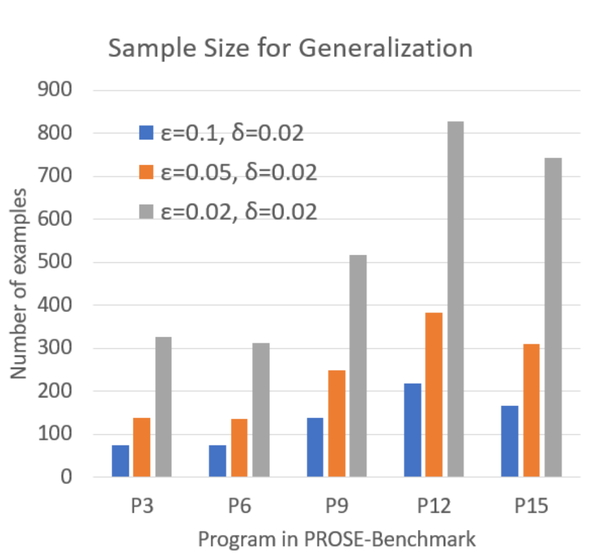
Evaluation

for **SynGuar + StrPROSE**

- Under 400 examples for all testcases to achieve (ϵ = 0.05, δ = 0.02).
- Improving correctness
  - ≈94% with SynGuar(0.05, 0.02)
  - ≈34% with 4 random examples

for **SynGuar + StrSTUN**

- Most of the testcases are under 500 examples for ϵ = 0.05, δ = 0.02.
- Improving correctness
  - ≈90% with SynGuar(0.05, 0.02)
  - ≈56% with 4 random examples
  - ≈61% with provided examples



Online Demo

GitHub