Guarantees in Program Synthesis

UNIVERSITY OF WISCONSIN–MADISON

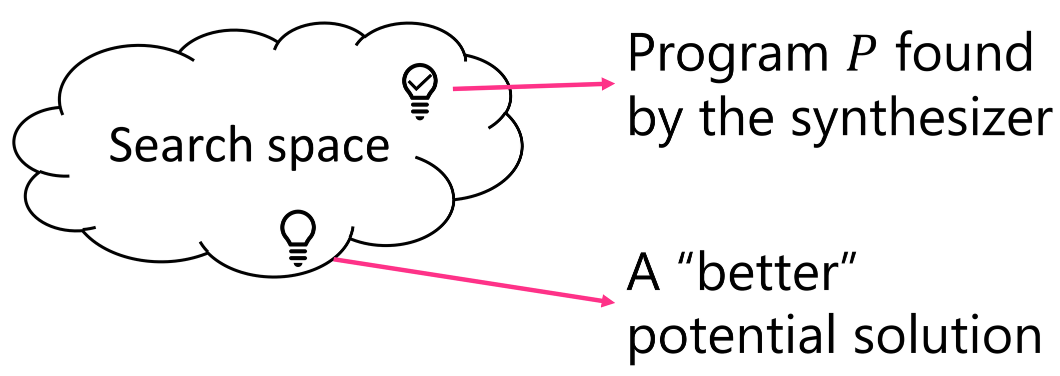
Qinheping Hu , Jason Breck , John Cyphert, Loris D'Antoni , Thomas Reps

madPL

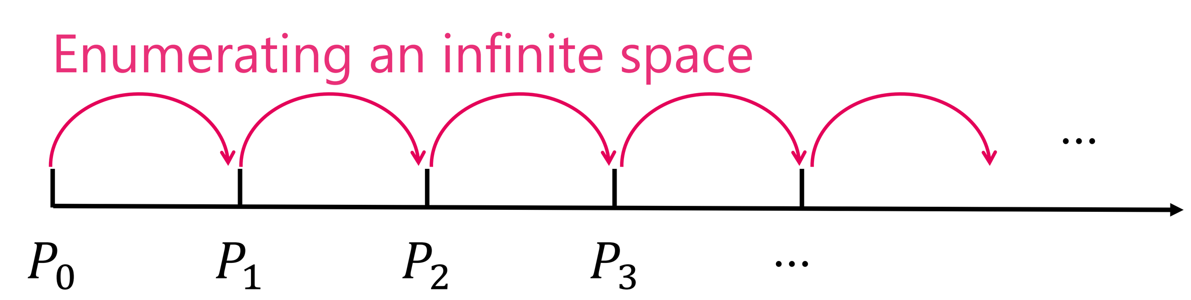
Introduction

Program synthesis is the classic problem of automatically finding a **program** P in some **search space** that satisfies a given correctness **specification**. Unfortunately, it is not always enough to produce any correct solution.

1. Synthesizers may return a worse solution.



2. Enumeration-based synthesizers have no ability to prove there is no solution in infinite search space.



Contributions

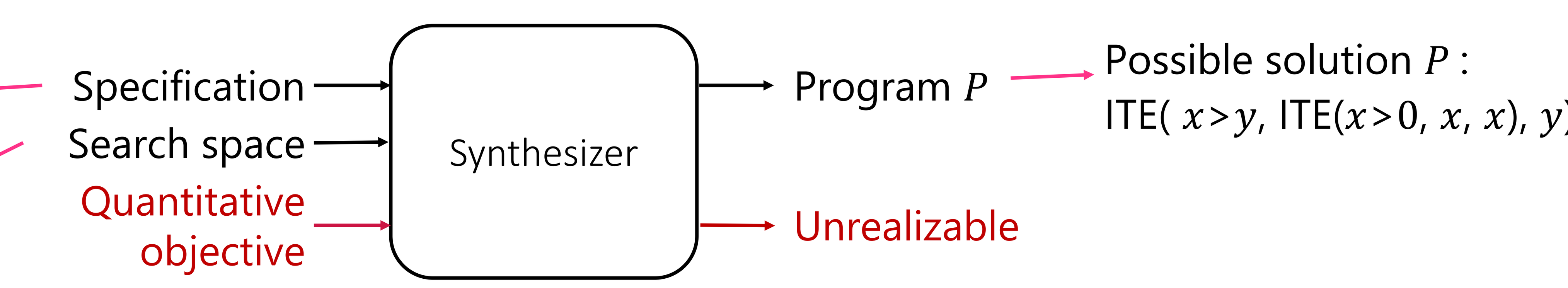
To address the above two problems, we introduce two types of guarantees in program synthesis: **quantitative objectives** and the ability to proof **unrealizable**—no solution.

Syntax-Guided Synthesis (SyGuS)

We illustrate the syntax-guided synthesis by an example of synthesizing a function computing the maximum of two integers.

A logic formula
$\phi(P)\colon \forall x, y. P \geq x \wedge P \geq y \wedge (P = x \vee P = y)$

A grammar G
Start := Start+Start | ITE(BExpr,Start,Start) | x | 0 | y | 1
BExpr := NOT(BExpr) | Start > Start | Start AND Start



Program synthesizer with more Guarantees

SyGuS + Quantitative Objective [CAV18]

QSyGuS

In SyGuS with Quantitive Objective (QSyGuS), users can assign weights to grammar productions and ask the solver to find solutions with minimized/maximized weights.
For example, if the user wants to minimize the number of ITE in the above SyGuS problem, he can assign weights to productions as follow

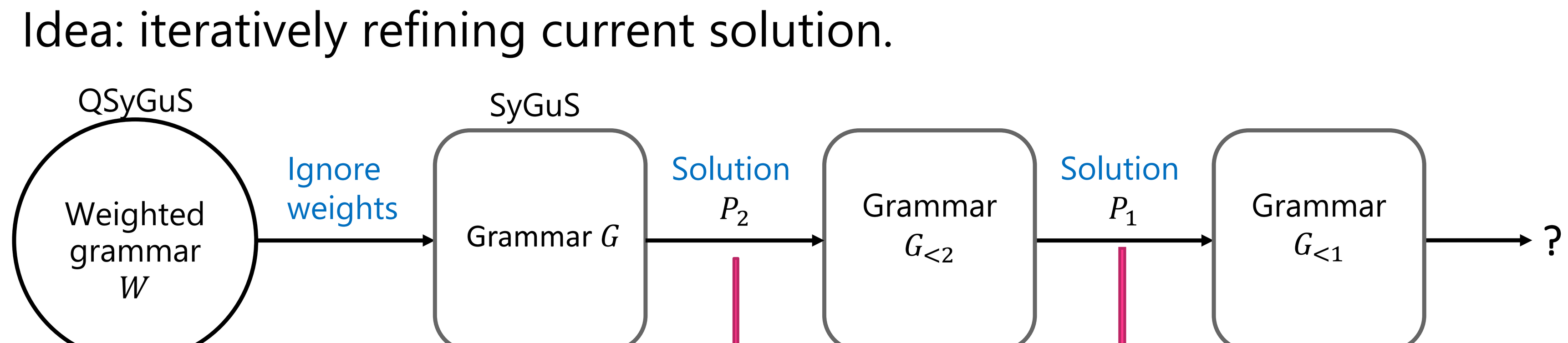
Weighted grammar W:

Start := Start+Start	0	BExpr := NOT(BExpr)	0
| ITE(BExpr,Start,Start)	1	| Start > Start	0
| x | 0 | y | 1	0	| Start AND Start	0

Observe that weights of terms in the above grammar is equal to the number of ITE in the terms.

Algorithm of finding optimized solution

Idea: iteratively refining current solution.



Solution P_2 with weight 2:
ITE($x>y$, ITE($x>0$, x, x), y)

Solution P_1 with weight 1:
ITE($x>y$, x, y)

Grammar $G_{<2}$ producing terms with weight less than 2:

Start := Start0 | Start1
Start1 := Start1+Start0
  | ITE(BExpr0,Start0,Start0)
  | x | y | 0 | 1
Start0 := Start1+Start0
  | x | y | 0 | 1
BExpr0 := NOT(BExpr0)
  | Start0 > Start0
  | Start0 AND Start0

Grammar $G_{<1}$:
Start := Start+Start
  | x | y | 0 | 1

> P_1 is optimal
> Iff $G_{<1}$ is unrealizable

Proving a SyGuS problem is Unrealizable [CAV19]

Observation

The problem of finding a solution P in $G_{<1}$ is already unrealizable for the following input examples

$$E\colon P(0,0) = 0, P(1,0) = 1, P(0,1) = 1, P(0,2) = 2$$

Idea

We reduce the problem of checking unrealizability based on the examples to the following verification problem that can be solved using off-the-shelf verifiers

```
int[4] Start(x_0,y_0,x_1,y_1,x_2,y_2,x_3,y_3){
     if(??){return (0,0,0,0);} // Start -> 0
     if(??){return (1,1,1,1);} // Start -> 1
     if(??){return (x_0,x_1,x_2,x_3);} // Start -> x
     if(??){return (y_0,y_1,y_2,y_3);} // Start -> y
     else{     // Start -> Start+Start
          int[4] L = Start(x_0,y_0,x_1,y_1);
          int[4] R = Start(x_0,y_0,x_1,y_1);
          return (L[0]+R[0],L[1]+R[1],L[2]+R[2],L[3]+R[3]);}
}
int[4] P = Start(0,0,0,1,1,0,2,0);
assert (P[0]!=0 || P[1]!=1 || P[2]!=1 || P[3]!=2);
```

> The SyGuS problem is unrealizable
> Iff the `assert` always holds

Future Work

- Other quantitative objectives in program synthesis:
  - Semantic quantitative objectives
  - Resource bounded synthesis
- Proving unrealizability for synthesis problems beyond SyGuS

[CAV18] Hu, D'Antoni. Syntax Guided Synthesis with Quantitative Syntactic Objectives
[CAV19] Hu, Breck, Cyphert, D'Antoni, Reps. Proving Unrealizability for Syntax-Guided Synthesis