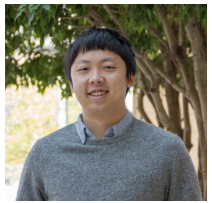
SYNTHESIS WITH GUARANTEES

Loris D'Antoni

University of Wisconsin Madison



Qinheping Hu

Loris D'Antoni

John Cyphert

Thomas Reps

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

madPL

Syntax Guided Synthesis [Alur et al. 13]

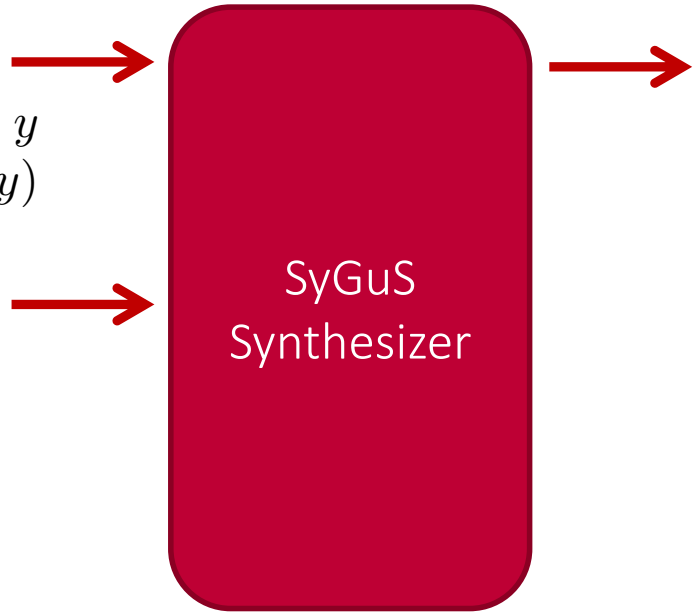
Specification

$$\Phi(P) : \forall x, y. P \geq x \wedge P \geq y \wedge (P = x \vee P = y)$$

Search space

Start := Start+Start
| ITE(BExpr,Start,Start)
| x | y | 0 | 1

BExpr := NOT(BExpr)
| Start > Start
| Start AND Start



Program

ITE(x>y, x, y)

ITE(x>y, ITE(x>0, x, x), y)

Applications of SyGuS

Automatic Program Inversion using Symbolic Transducers

Univer

Quantified Invariants via
Syntax-Guided Synthesis

Synthesis of Fault-Attack Countermeasures for Cryptographic Circuits

[1] Princ
[2] TC

Accelerating Syntax-Guided Invariant Synthesis

Grigory Fedyukovich[1] and Rastislav Bodík[2]

[1] Princeton University, USA, grigoryf@cs.princeton.edu
[2] University of Washington, USA, bodik@cs.washington.edu

DOES IT REALLY WORK

```
1   // Axuiliary functions
2   fun E (x: x <= #x40) :=
3      (ite (x <= #x19) (x + #x41)
4         (ite (x <= #x33) (x + #x47)
5           (ite (x <= #x3d) (x - #x04)
6             (ite (x == #x3e) #x2b #x2f))))
7   fun B h l x := (x << (7 - h)) >> (7 - h + l)
8   // List transformations
9   trans B64E (l: (BitVec 8) list) : (BitVec 8) :=
10     match l with
11     | x::y::z::tail when true ->
12       E (B 7 2 x) ::
13       E (((B 1 0 x) << #x04) | (B 7 4 y)) ::
14       E (((B 4 0 y) << 2) | (B 7 6 z)) ::
15       E (B 5 0 z) :: B64E(tail)
16     | x::y::[] when true ->
17       E (B 7 2 x) ::
18       E (((B 1 0 x) << 4) | (
19       E ((B 4 0 y) << 2) :: #
20     | x::[] when true ->
21       E (B 7 2 x)) ::
22       E ((B 1 0 x) << 4) :: #x3
23     | [] when true -> []
```

CVC4

ESolver

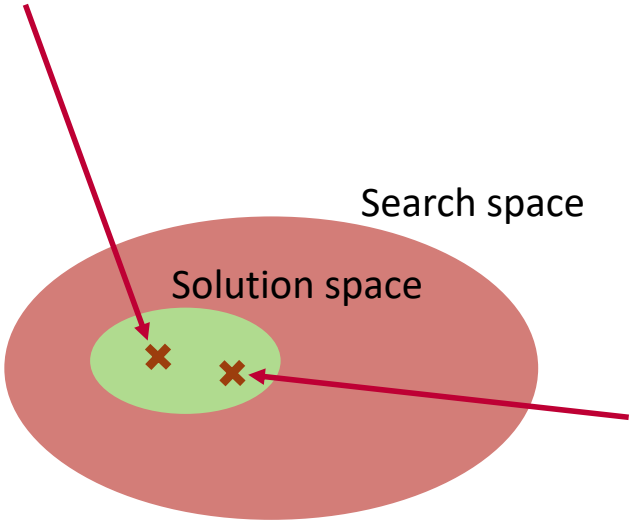
```
(define-fun ((x (BitVec 8)) (y (BitVec 8))) (bvand (bvlshl (DD x) #x02) (bv
```

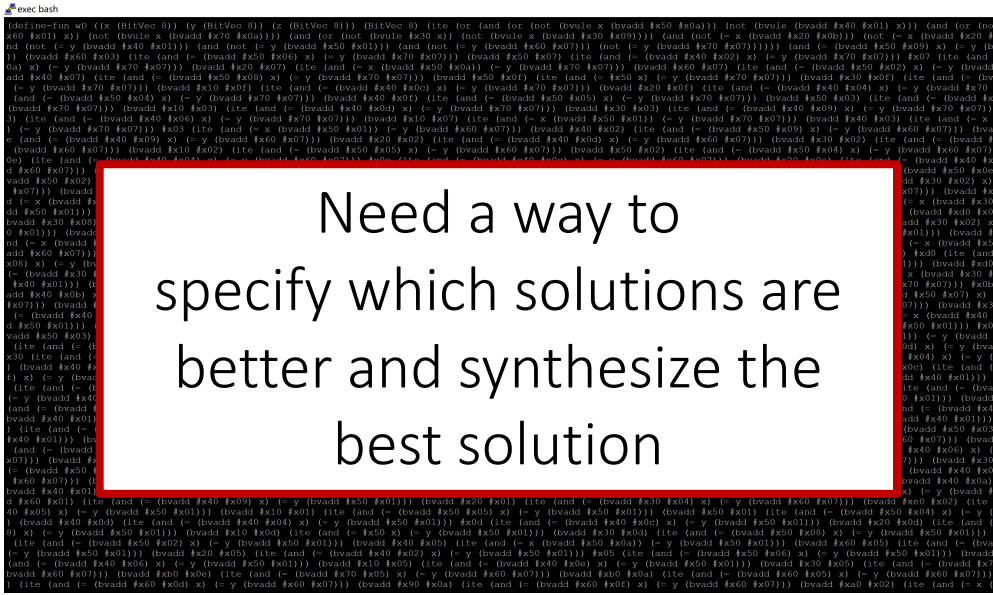
exec bash

```
(define-fun wD ((x (BitVec 8)) (y (BitVe
x60 #x01) x)) (not (bvule x (bvadd #x70
nd (not (= y (bvadd #x40 #x01))) (and (n
)) (bvadd #x60 #x03) (ite (and (= (bvadd
0a) x) (= y (bvadd #x70 #x07))) (bvadd #
add #x40 #x07) (ite (and (= (bvadd #x50
 (= y (bvadd #x70 #x07))) (bvadd #x10 #x
 (and (= (bvadd #x50 #x04) x) (= y (bvad
(bvadd #x70 #x07))) (bvadd #x10 #x03) (i
3) (ite (and (= (bvadd #x40 #x06) x) (=
) (= y (bvadd #x70 #x07))) #x03 (ite (an
e (and (= (bvadd #x40 #x09) x) (= y (bva
 (bvadd #x60 #x07))) (bvadd #x10 #x02) (
0e) (ite (and (= (bvadd #x40 #x04) x) (=
d #x60 #x07))) (bvadd #x10 #x0e) (ite (a
vadd #x50 #x02) x) (= y (bvadd #x60 #x07
 #x07))) (bvadd #xd0 #x0a) (ite (and (=
d (= x (bvadd #x30 #x03)) (= y (bvadd #x
dd #x50 #x01))) (bvadd #xf0 #x05) (ite (
bvadd #x30 #x08) x) (= y (bvadd #x50 #x0
0 #x01))) (bvadd #xd0 #x09) (ite (and (=
nd (= x (bvadd #x30 #x03)) (= y (bvadd #
add #x60 #x07))) (bvadd #x60 #x06) (ite
x08) x) (= y (bvadd #x40 #x01))) #xf0 (i
(= (bvadd #x30 #x06) x) (= y (bvadd #x40
 #x40 #x01))) (bvadd #xd0 #x0c) (ite (an
add #x40 #x0b) x) (= y (bvadd #x70 #x07)
#x07))) (bvadd #x50 #x0b) (ite (and (=
 (= (bvadd #x40 #x08) x) (= y (bvadd #x4
d #x50 #x01))) (bvadd #x10 #x09) (ite (a
vadd #x50 #x03) x) (= y (bvadd #x50 #x01
 (ite (and (= (bvadd #x50 #x09) x) (= y
x30 (ite (and (= (bvadd #x40 #x05) x) (=
) (bvadd #x40 #x0c) (ite (and (= (bvadd
f) x) (= y (bvadd #x50 #x01))) (bvadd #x
 (ite (and (= (bvadd #x50 #x02) x) (= y
(= y (bvadd #x40 #x01))) (bvadd #x20 #x0
(and (= (bvadd #x40 #x06) x) (= y (bvadd
bvadd #x40 #x01))) (bvadd #x30 #x08) (it
) (ite (and (= (bvadd #x40 #x0b) x) (= y
#x40 #x01))) (bvadd #x40 #x08) (ite (and
 (and (= (bvadd #x40 #x02) x) (= y (bvad
x07))) (bvadd #x10 #x06) (ite (and (= (b
(= (bvadd #x50 #x07) x) (= y (bvadd #x60
 #x60 #x07))) (bvadd #x20 #x0a) (ite (an
bvadd #x40 #x01)) (= y (bvadd #x60 #x07)
d #x60 #x01) (ite (and (= (bvadd #x40 #x
40 #x05) x) (= y (bvadd #x50 #x01))) (bv
) (bvadd #x40 #x0d) (ite (and (= (bvadd
8) x) (= y (bvadd #x50 #x01))) (bvadd #x
 (ite (and (= (bvadd #x50 #x02) x) (= y
(= y (bvadd #x50 #x01))) (bvadd #x20 #x0
(and (= (bvadd #x40 #x06) x) (= y (bvadd
bvadd #x60 #x07))) (bvadd #xb0 #x0e) (it
) (ite (and (= (bvadd #x60 #x0d) x) (= y
```

Program synthesis is unpredictable – part 1

```
(define-fun ((x (BitVec 8)) (y (BitVec 8))) (bvand (bvlshl (DD x) #x02) (bvlshr (DD y) #x06)))
```

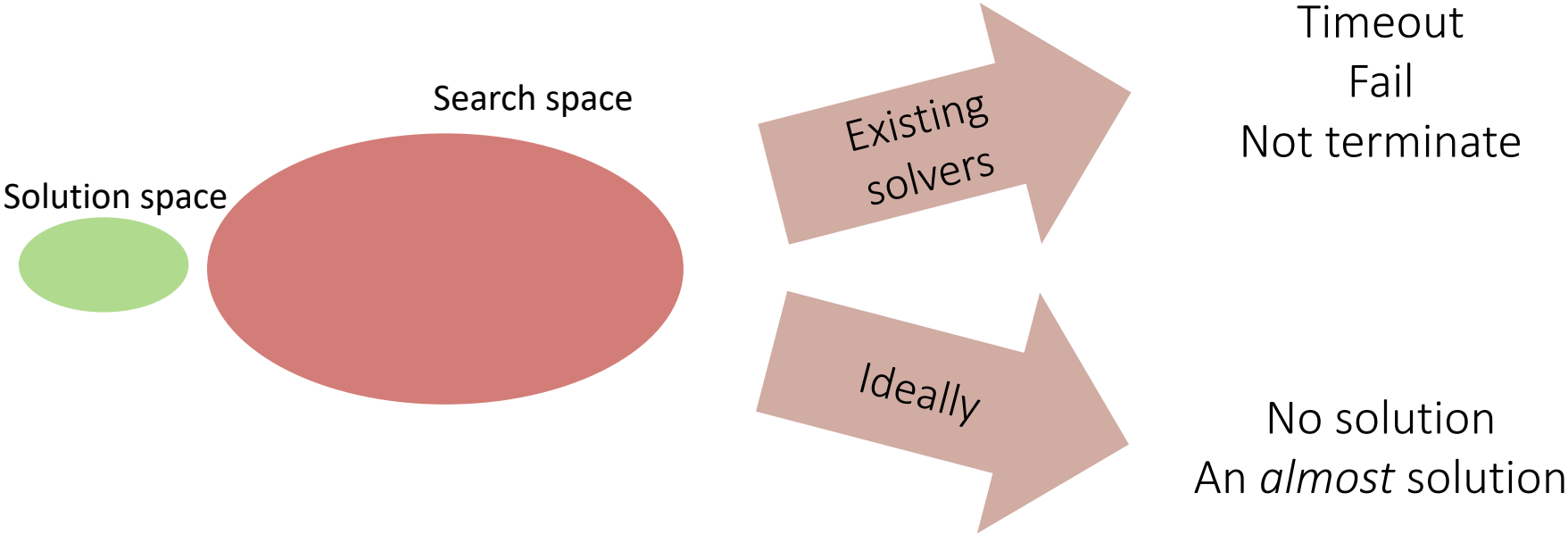




Need a way to specify which solutions are better and synthesize the best solution

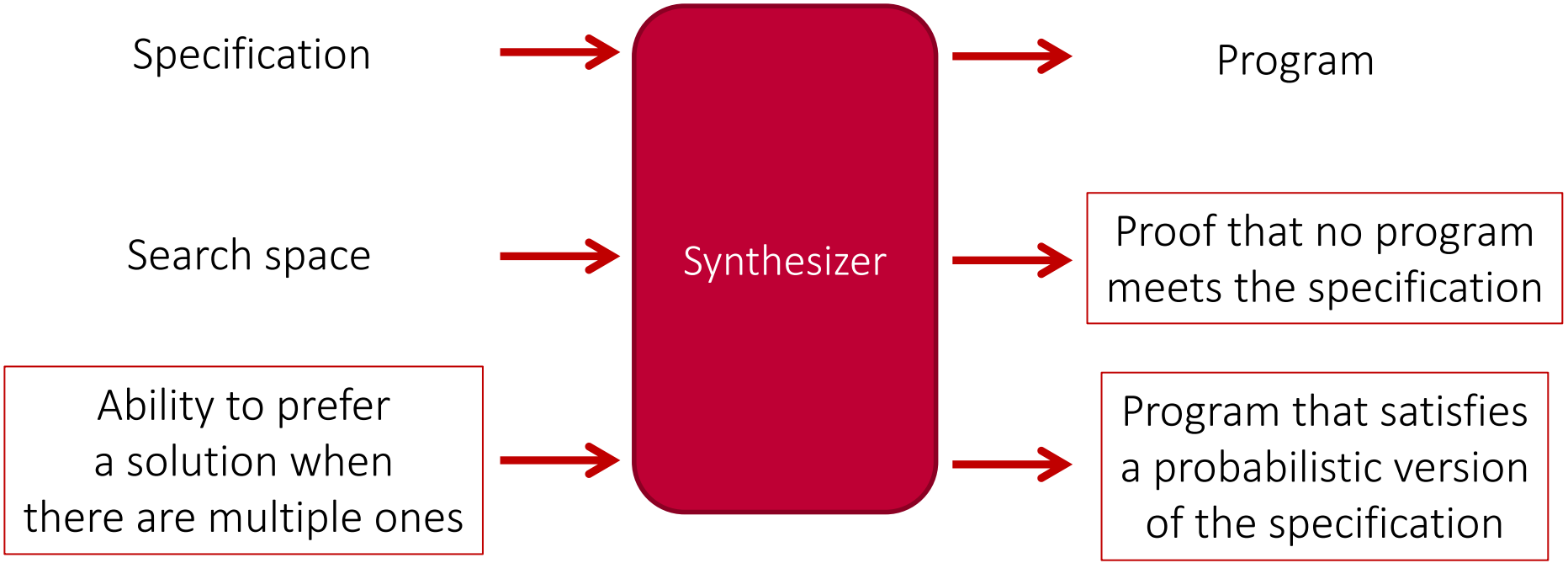
“Synthesis is like a box of chocolate,
You never know what you’re gonna get”

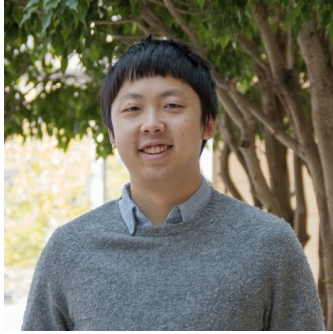
Program synthesis is unpredictable – part 2



PROGRAM SYNTHESIS WITH GUARANTEES

Program synthesis with guarantees





Qinheping Hu

SYNTAX-GUIDED SYNTHESIS WITH QUANTITATIVE SYNTACTIC OBJECTIVES

[CAV18]

Adding quantitative objectives

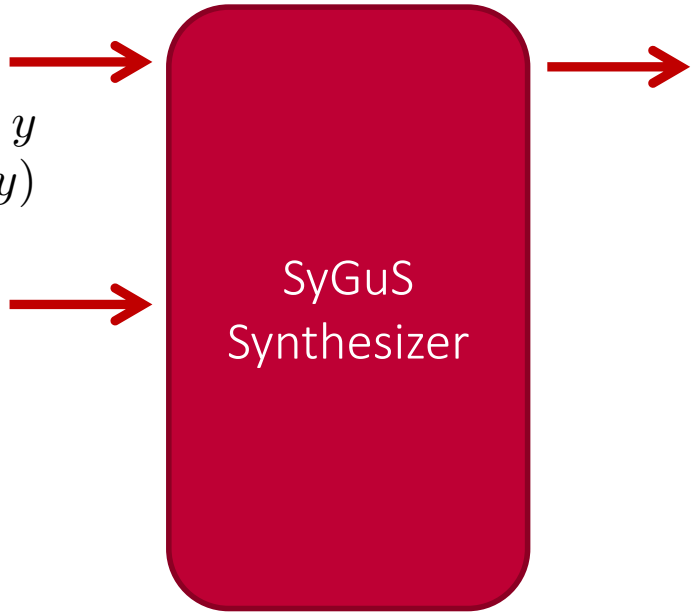
Specification

$$\Phi(P) : \forall x, y.P \geq x \wedge P \geq y \wedge (P = x \vee P = y)$$

Search space

Start := Start+Start
| ITE(BExpr,Start,Start)
| x | y | 0 | 1

BExpr := NOT(BExpr)
| Start > Start
| Start AND Start



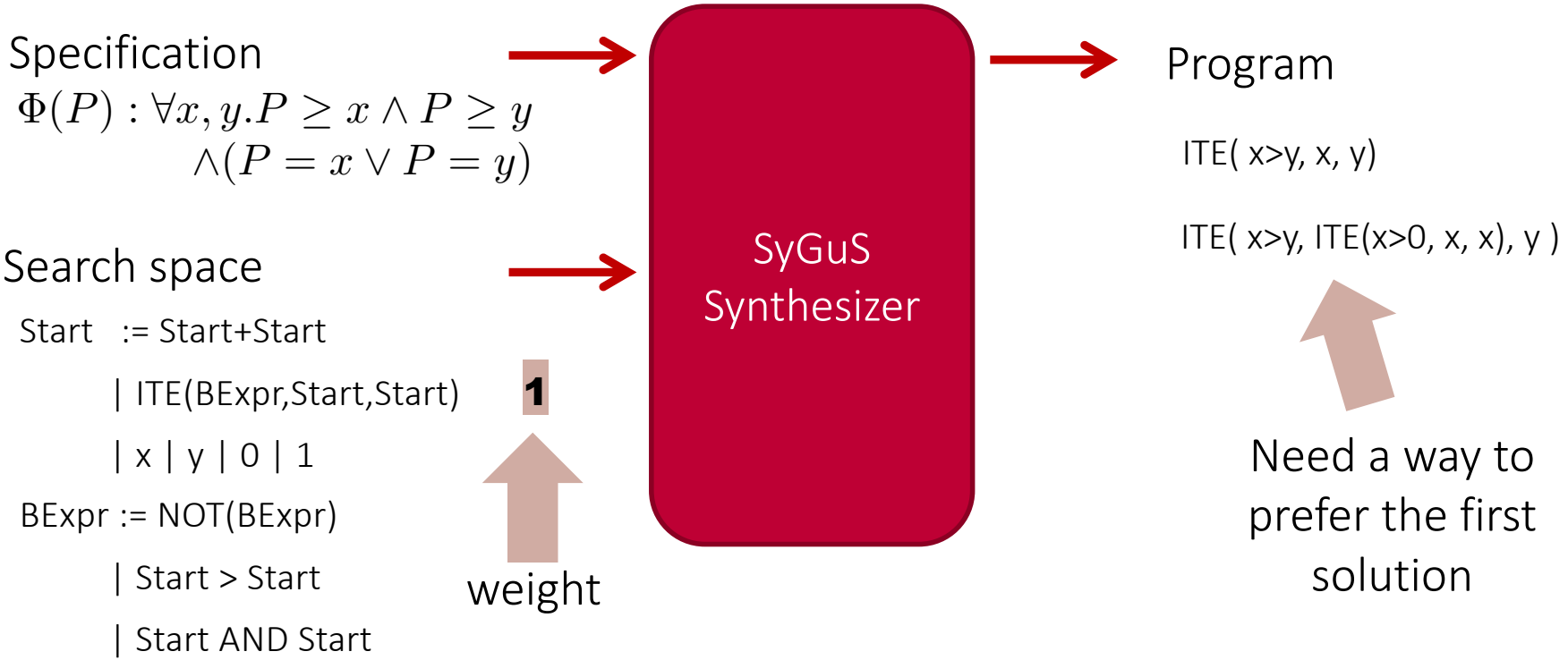
Program

ITE(x>y, x, y)

ITE(x>y, ITE(x>0, x, x), y)

Need a way to prefer the first solution

Adding quantitative objectives



What is the weight of a program?

Start := Start+Start

| ITE(BExpr,Start,Start) **1**

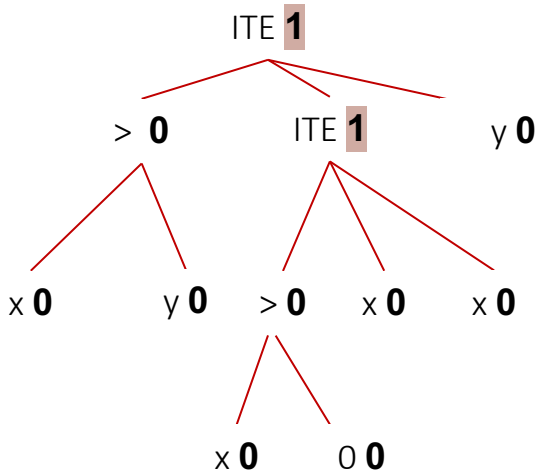
| x | y | 0 | 1

BExpr := NOT(BExpr)

| Start > Start

| Start AND Start

ITE(x>y, ITE(x>0, x, x), y) weight=2



Adding quantitative objectives

Specification

$$\Phi(P) : \forall x, y.P \geq x \wedge P \geq y \wedge (P = x \vee P = y)$$

Search space

Start := Start+Start

| ITE(BExpr,Start,Start) **1**

| x | y | 0 | 1

BExpr := NOT(BExpr)

| Start > Start

| Start AND Start

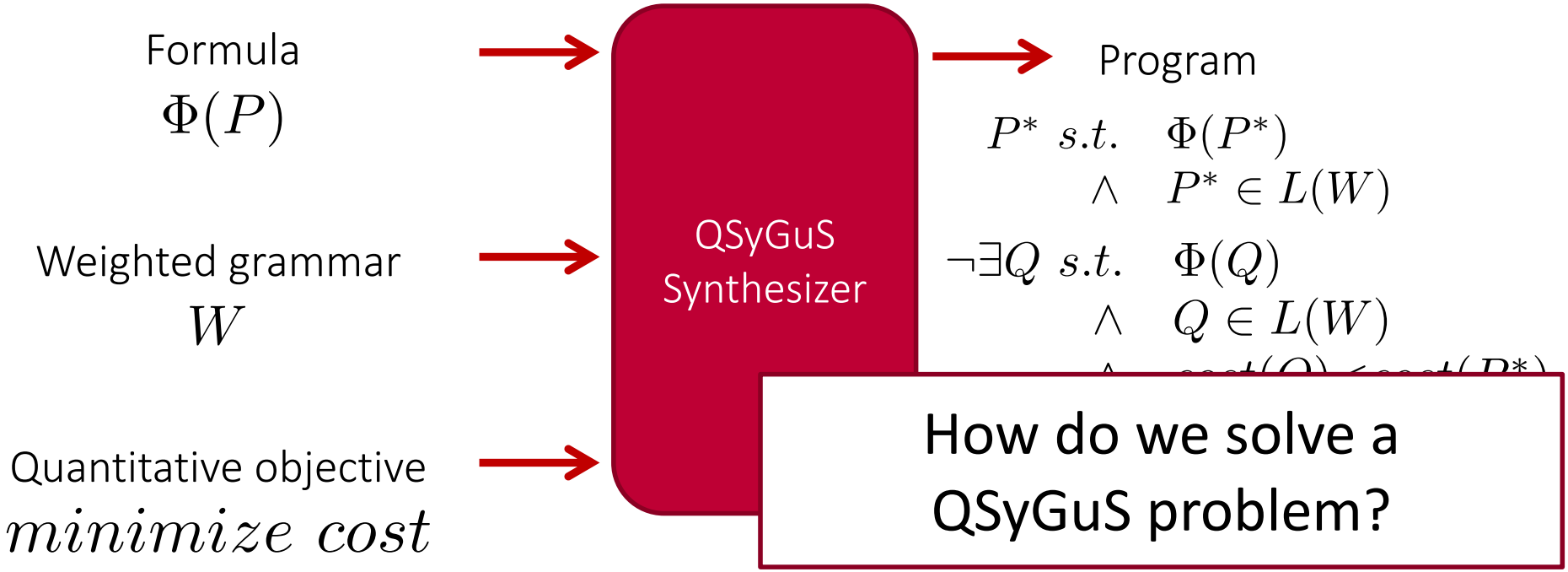
Program

ITE(x>y, x, y) **1**

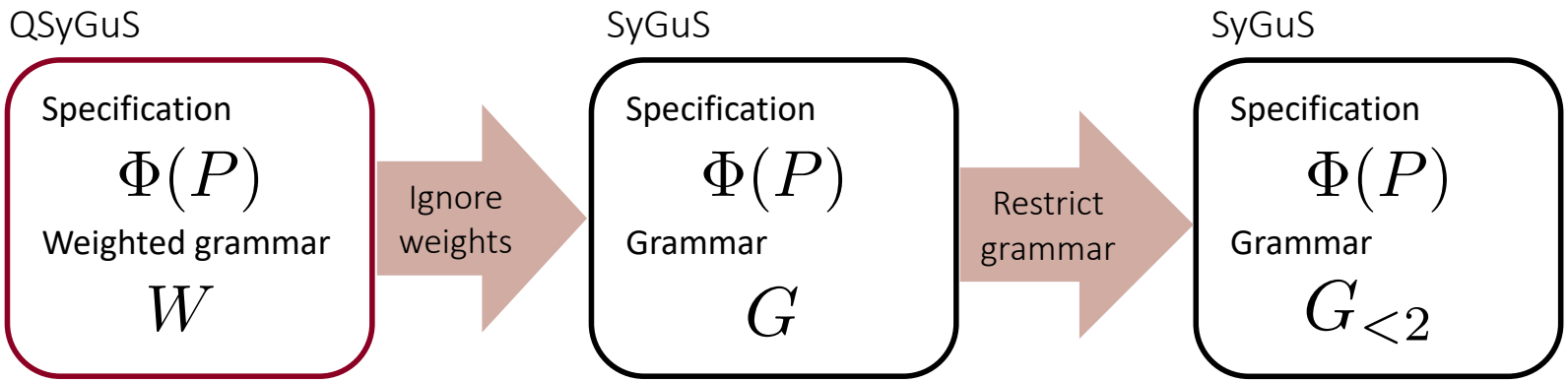
ITE(x>y, ITE(x>0, x, x), y) **2**

Programs now have weights/costs

Syntax Guided Synthesis with Quantitative Objectives



Solving QSyGuS problems



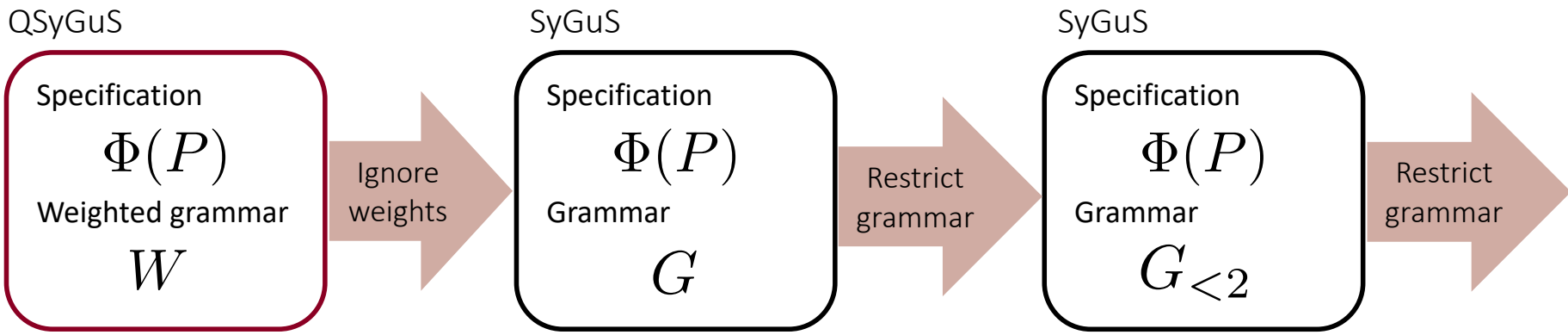

```
Start   := Start+Start
        | ITE(BExpr,Start,Start)
        | x | y | 0 | 1
…
```

1

```
Start   := Start+Start
        | ITE(BExpr,Start,Start)
        | x | y | 0 | 1
…
```

ITE(x>y, ITE(x>0, x, x), y) **2**

Solving QSyGuS problems



Start := Start+Start
| ITE(BExpr,Start,Start) **1**
| x | y | 0 | 1
…

ITE(x>y, ITE(x>0, x, x), y) **2**

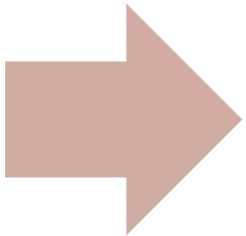
Start := Start**0** | Start**1**
Start**0** := Start**0**+Start**0**
| x | y | 0 | 1
Start**1** := ITE(BExpr, Start**0**, Start**0**)
| Start**0**+Start**1** | Start**1**+Start**0**

ITE(x>y, x, y) **1**
…

Soundness of grammar restriction

Weighted grammar does not contain negative weights

W



Reduced grammar accepts all and only the terms of weight < c

$G_{<c}$

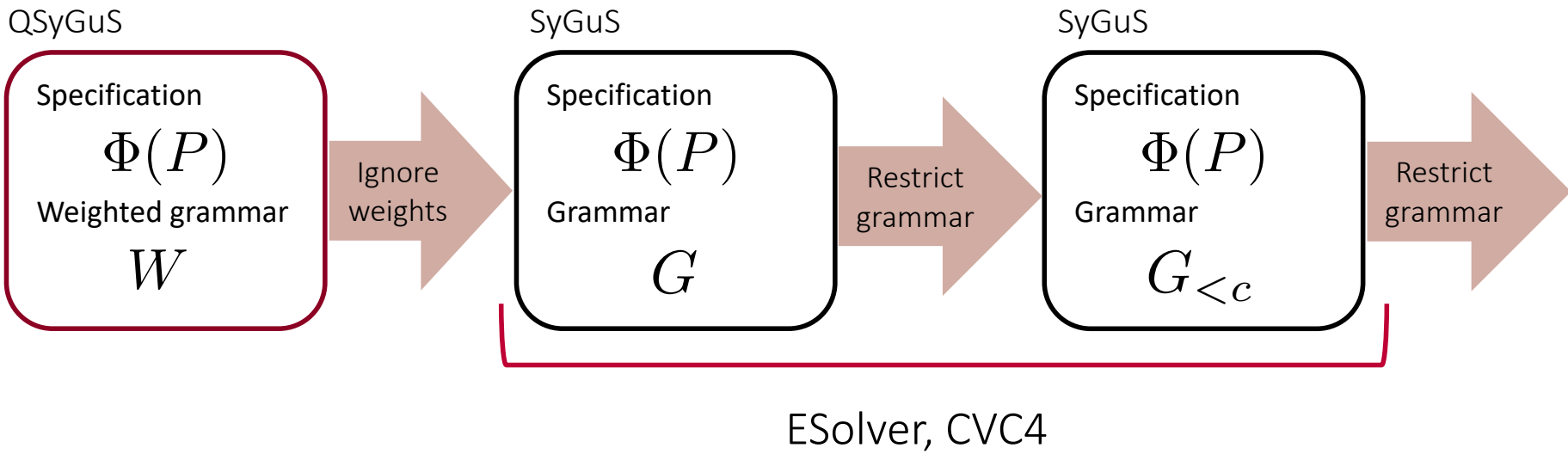
Results also generalize to multiplicative weights and bounded negative weights

Beyond minimization constraints

$minimize\ cost$	Linear search
$cost > 3$	$complement(G_{<4})$
$2 < cost < 5$	$G_{>2} \cap G_{<5}$
$3 < cost_1 \wedge cost_2 < 0.5$	$G_{cost_1>3} \cap G_{cost_2<0.5}$

DOES IT WORK?

Implementation



Benchmarks

26 SyGuS benchmarks

$(\mathbb{R}, +)$: minimize number of specified operator
minimize solution size

$([0,1], *)$: maximize solution probability

$(\mathbb{R}, +) \times (\mathbb{R}, +)$: find sorted optimal for (# of specified operators, size)
find Pareto optimal for (# of specified operators, size)

	Problem
Trop	max_ite(2,3)
	max_ite(2,15)
	max_ite(3,15)
	max_ite(10,15)
	parity_not
	max3_ite
	array_search_3
	array_search_5
	hackers_5
	hackers_7
	hackers_17
	hackers_19
	icfp_7
	LinExpr_eq1ex
Prob	hackers_2_prob
	hackers_5_prob
	hackers_7_prob
	hackers_17_prob

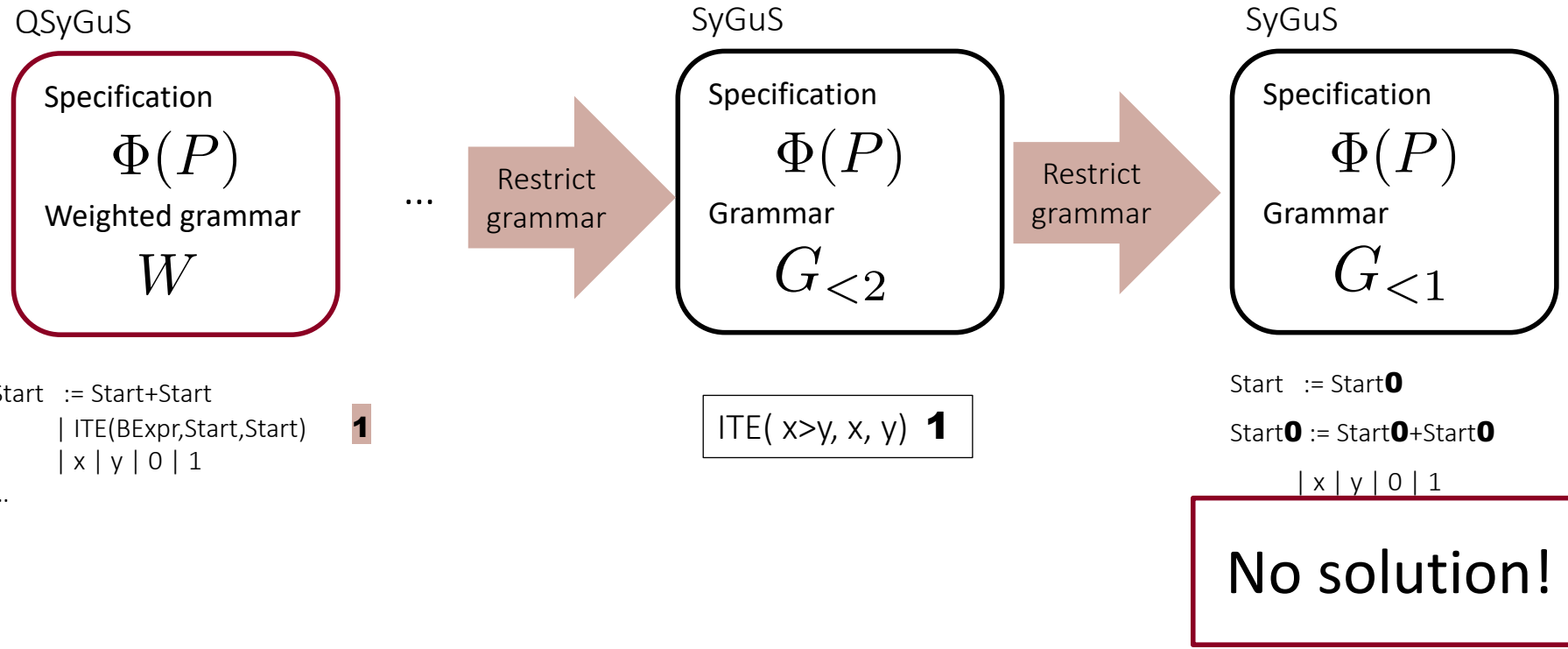
	Problem
Trop × Trop	array_search_sorted
	hackers_5_sorted
	hackers_7_sorted
	hackers_17_sorted
	array_search_pareto
	hackers_5_pareto
	hackers_7_pareto
	hackers_17_pareto

Summary of results

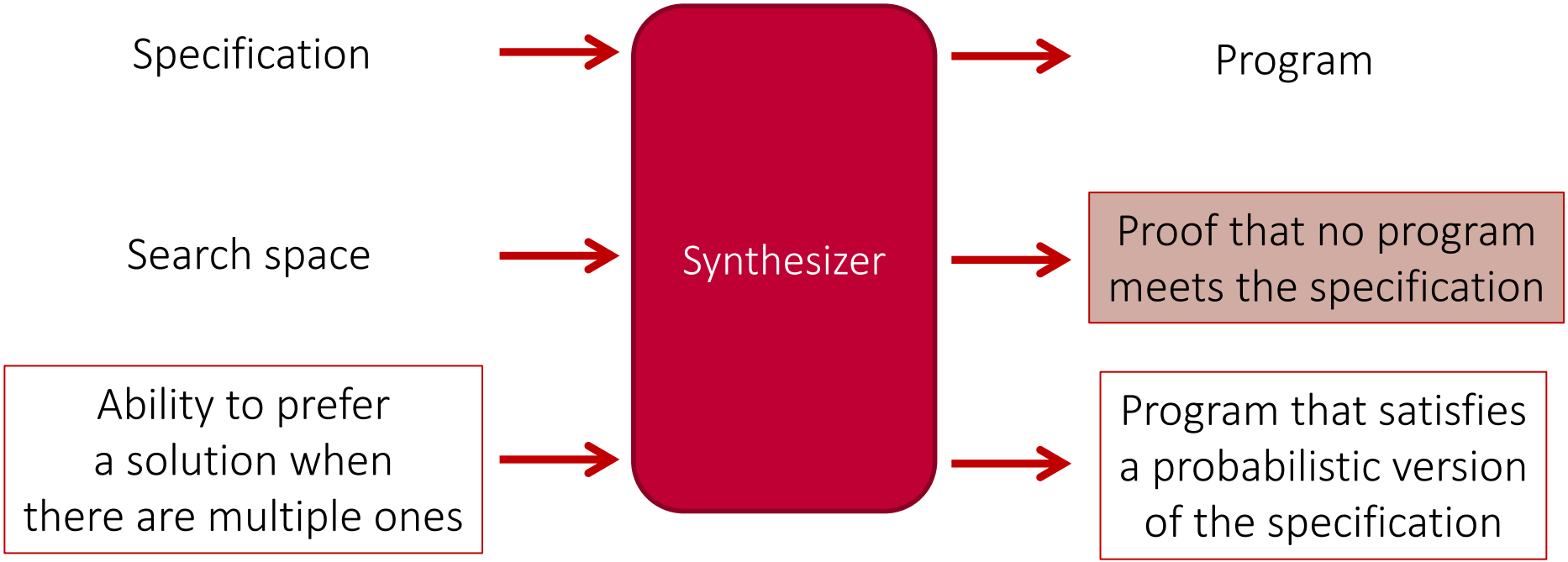
Solution with **better cost** than one without QSyGuS for **16/26** benchmarks

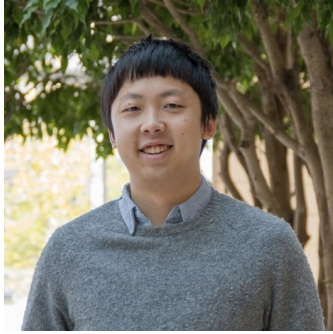
Found **optimal** solution for **14/26 benchmarks**

Why couldn't we prove optimality?



Program synthesis with guarantees

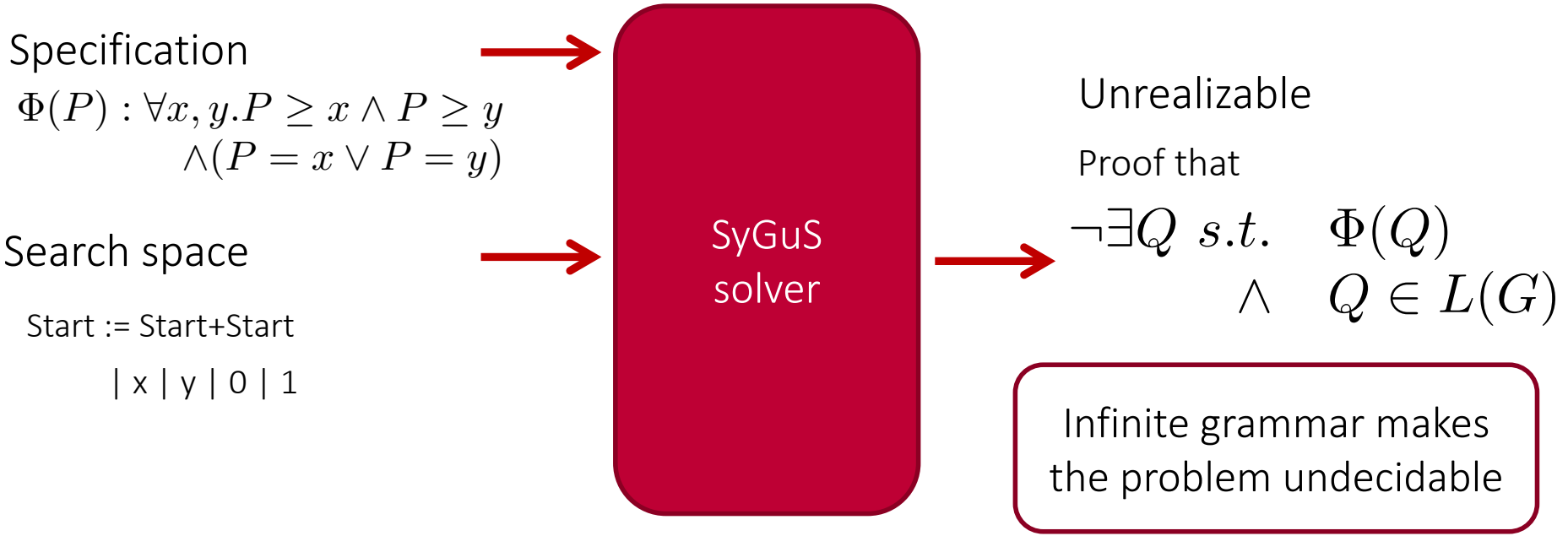




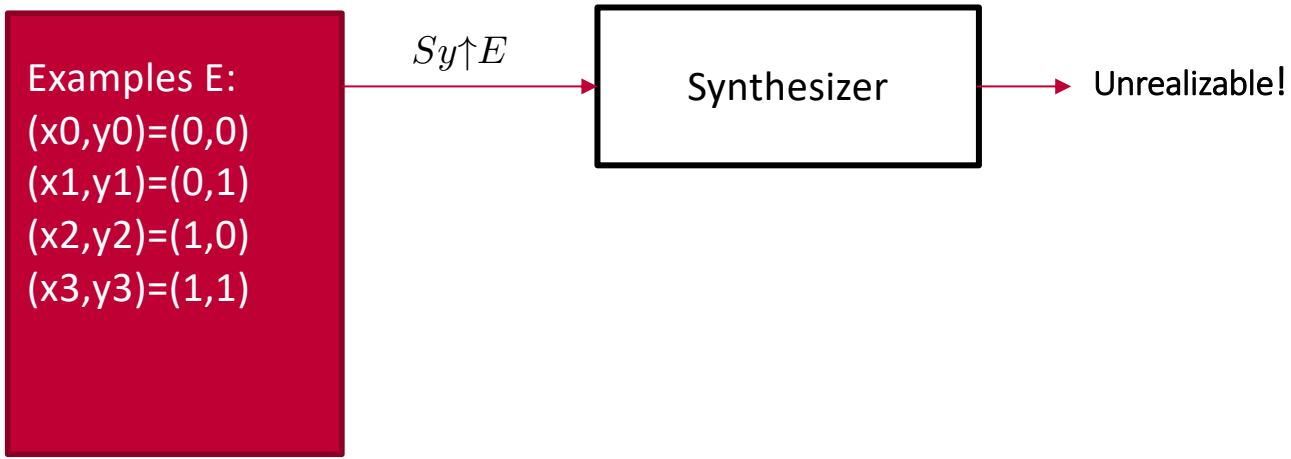
PROVING UNREALIZABILITY IN SYNTAX-GUIDED SYNTHESIS

Q. HU, J. BRECK, J. CYPHERT, L. D'ANTONI, T. REPS [CAV19]

Why is this hard?



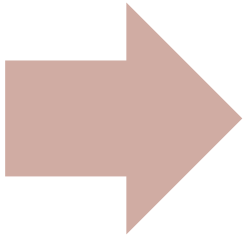
$$Sy{\uparrow}E = \wedge_{(x,y)\in E}\ \Phi(f, x, y)$$



Soundness of CEGIS for unrealizability

$Sy{\uparrow}E$ unrealizable

No solution over E

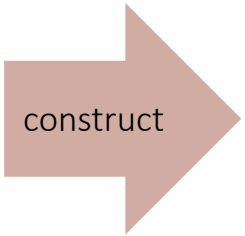


Sy unrealizable

No solution

Outline of the algorithm

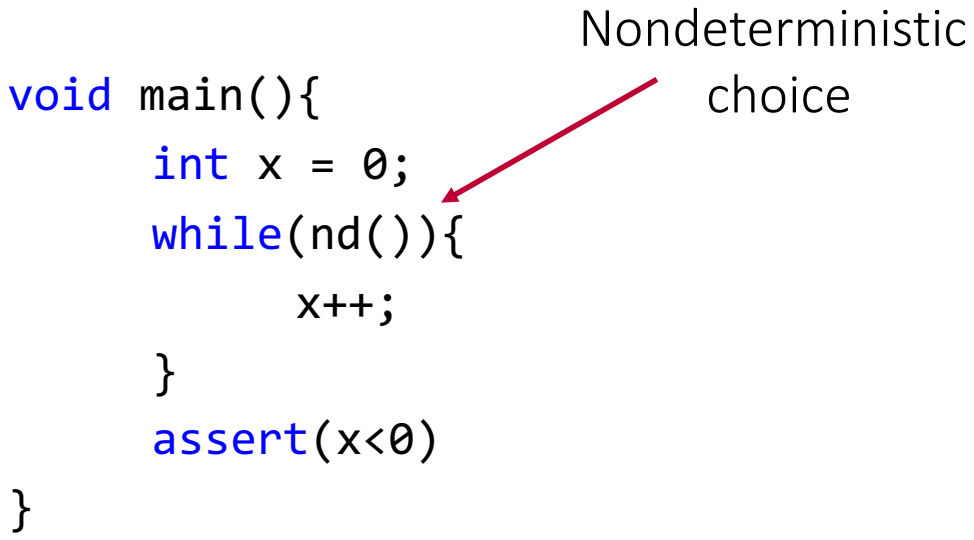
$Sy{\uparrow}E := (\Phi, G, E)$




```
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);
```

$Sy{\uparrow}E$ unrealizable  `assert` always holds

Reachability Problem



Reachability solvers:

CPA-checker

Uautomizer

Seahorn

Goal: can the `assert` be falsified?

Sy^E to Re^E

Set input to E

$$\vec{x} \leftarrow E$$

f_G is non-deterministically drawn from $L(G)$

$$\vec{o} \leftarrow f_G(\vec{x})$$

Check if $\vec{o}$ doesn't satisfy φ ⟺ $f_G(\vec{x})$ does not satisfy φ on E

$$\texttt{assert}(\ \neg\varphi(o, x)\ , x_i))$$

$Sy{\uparrow}E$ unrealizable

Sy^E to Re^E

Set input to E

$$\vec{x} \leftarrow E$$

f_G is non-deterministically drawn from $L(G)$

$$\vec{o} \leftarrow f_G(\vec{x})$$

Check if $\vec{o}$ doesn't satisfy φ

$$\texttt{assert}(\quad \neg\varphi(o, x) \quad , x_i))$$

Check if $\vec{o}$ doesn't satisfy φ

$$\texttt{assert}(\neg \bigwedge x_i \in E.\, \varphi(o_i, x_i))$$

```
void main(){
    …
    assert(!(spec(x0,y0,o0)&&spec(x1,y1,o1)));
}
bool spec(x,y,o){
    return (o>=x)&&(o>=y)&&(o==x||o==y);
}
```

Sy^E to Re^E

Set input to E

$$\vec{x} \leftarrow E$$

f_G is non-deterministically drawn from $L(G)$

$$\vec{o} \leftarrow f_G(\vec{x})$$

Check if $\vec{o}$ doesn't satisfy φ

$$\texttt{assert}(\quad \neg\varphi(o, x) \quad , x_i))$$

f_G is non-deterministically drawn from $L(G)$

$$\vec{o} \leftarrow f_G(\vec{x})$$

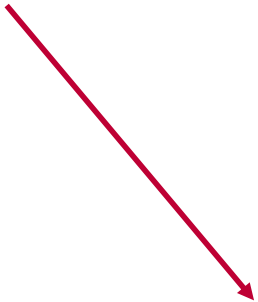
```
o0 = fStart(x0,y0);

int fStart(x0,y0){          \\ Start -> 0
   if(nd()){ return 0;}     \\ Start -> 1
   if(nd()){ return 1;}     \\ Start -> x
   if(nd()){ return x0;}    \\ Start -> y
   if(nd()}{ return y0;}    \\ Start -> +(Start,Start)
   if(nd()){
      left = fStart(x0,y0);
      right = fStart(x0,y0);
      return left + right;}
}
```

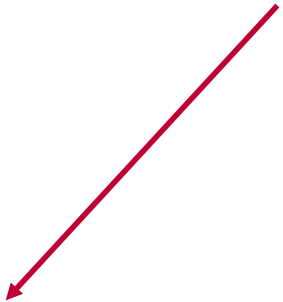
Example 0

o0=fStart(x0,y0);
o0 is f_G(x0,y0)for some f_G in $L(G)$

Example 1

o1=fStart(x1,y1);
o1 is f_G(x1,y1)for some f_G in $L(G)$





The two f_G can be different!

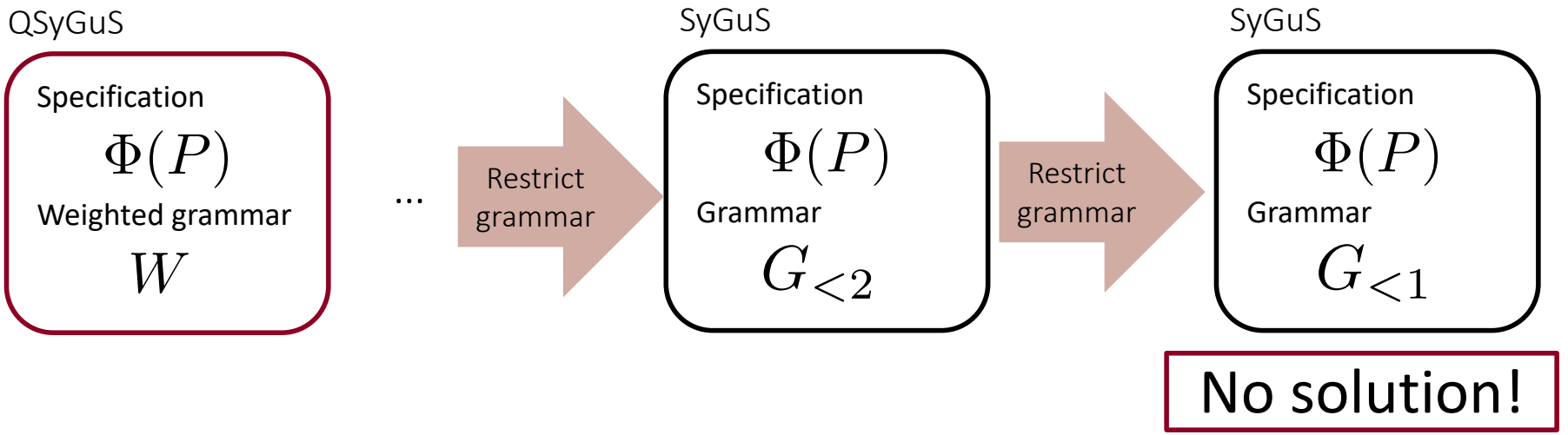
f_G is non-deterministically drawn from $L(G)$

$$\vec{o} \leftarrow f_G(\vec{x})$$

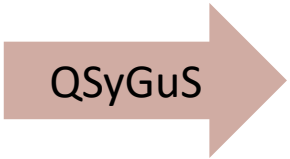
```
(o0,o1) = fStart(x0,y0,x1,y1);

<int,int> fStart(x0,y0,x1,y1){
   if(nd()){ return (0,0);}    \\ Start -> 0
   if(nd()){ return (1,1);}    \\ Start -> 1
   if(nd()){ return (x0,x1);} \\ Start -> x
   if(nd()}{ return (y0,y1);} \\ Start -> y
   if(nd()){                   \\ Start -> +(Start,Start)
      (a0,a1) = fStart(x0,y0,x1,y1);
      (b0,b1) = fStart(x0,y0,x1,y1);
      return (a0+b0,a1+b1);}
}
```

DOES IT WORK?

Benchmarks



60 SyGuS benchmarks over LIA



132 benchmarks that should be unrealizable

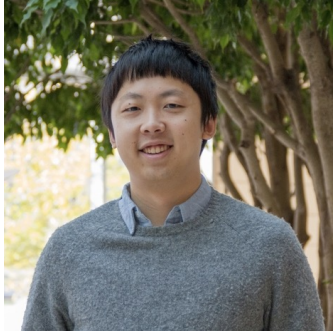
The tool Nope

132 variants of benchmarks taken from SyGuS	Solved
1. bounded number of if-operators	13/57
2. bounded number of plus-operators	1/30
3. restricted range of constants	45/45
	59/132

Limitation NOPE: size of grammars

	number of nonterminals	number of productions	number of examples	total time (s)	SeaHorn time (s)
mpg_example1	59	815	1	✗	✗
mpg_example2	21	178	1	✗	✗
mpg_example3	143	4186	1	✗	✗
mpg_example4	443	36745	1	✗	✗

Large sized reachability problem



Qinheping Hu

A SPECIALIZED UNREALIZABILITY TECHNIQUE

EXACT AND APPROXIMATE METHODS FOR PROVING UNREALIZABILITY OF SYNTAX-GUIDED SYNTHESIS PROBLEMS [PLDI20]

Example of an Unrealizable Problem

$$f(1) = 5$$
$$f(2) = 6$$

$\text{Start} \rightarrow \text{Expr}_1 \mid \text{Expr}_2$

Solution $\in$? $\text{Expr}_1 \rightarrow x + x + \text{Expr}_1 \mid 1$

Solution $\in$? $\text{Expr}_2 \rightarrow x + x + x + \text{Expr}_2 \mid 0$

Example of an Unrealizable Problem

$$f(1) = 5$$
$$f(2) = 6$$

$$\exists \lambda.\, 2\lambda 1 + 1 = 5$$
$$\wedge\, \underbrace{2\lambda 2 + 1}_{\text{odd}} = \underbrace{6}_{\text{even}}$$

$\text{Start} \rightarrow \text{Expr}_1 \mid \text{Expr}_2$

Solution $\in$? $\text{Expr}_1 \rightarrow x + x + \text{Expr}_1 \mid 1$

Solution $\in$? $\text{Expr}_2 \rightarrow x + x + x + \text{Expr}_2 \mid 0$

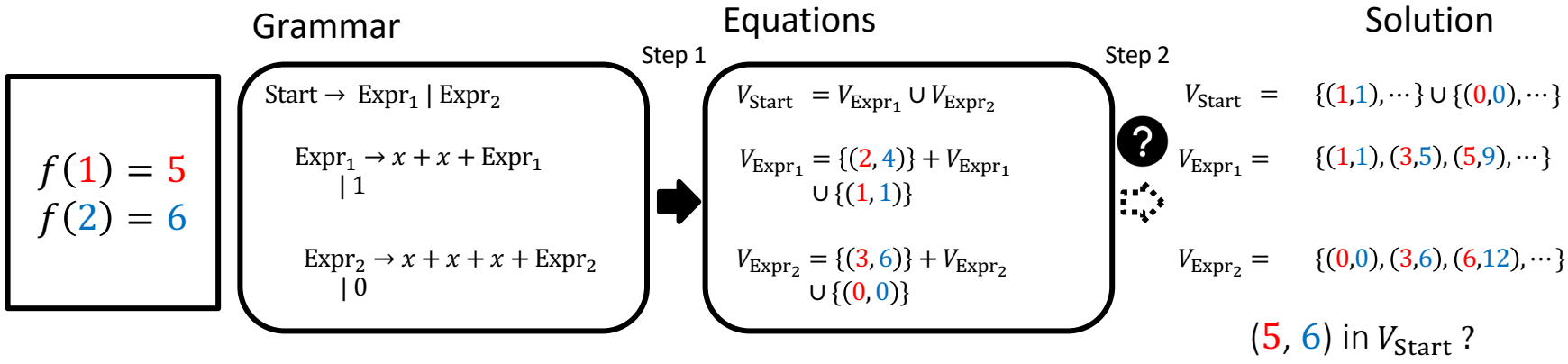
$\boldsymbol{2\lambda x + 1}$: $1, 2x + 1, 4x + 1, \ldots$

Example of an Unrealizable Problem

$f(1) = 5$
$f(2) = 6$

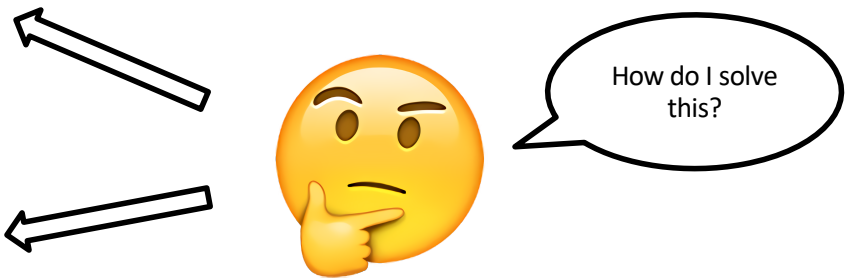
	$\text{Start} \rightarrow \text{Expr}_1 \mid \text{Expr}_2$	$\boldsymbol{2\lambda x + 1}$ or $\boldsymbol{3\lambda x}$
Solution 1 ✗	$\text{Expr}_1 \rightarrow x + x + \text{Expr}_1 \mid 1$	$\boldsymbol{2\lambda x + 1}$
Solution 2 ✗	$\text{Expr}_2 \rightarrow x + x + x + \text{Expr}_2 \mid 0$	$\boldsymbol{3\lambda x}$

High-level Idea



Logical approach: Constrained Horn Clauses (CHC)

Iterative approach



Logical approach: Solving Equations with Horn Clauses

$f(1) = 5$
$f(2) = 6$

Equations

$$V_{\text{Start}} = V_{\text{Expr}_1} \cup V_{\text{Expr}_2}$$

$$V_{\text{Expr}_1} = \{(2,4)\} + V_{\text{Expr}_1} \cup \{(1,1)\}$$

$$V_{\text{Expr}_2} = \{(3,6)\} + V_{\text{Expr}_2} \cup \{(0,0)\}$$

Horn Clauses

$$\forall x.\, x \in V_{\text{Start}} \leftarrow x \in V_{\text{Expr}_1} \vee x \in V_{\text{Expr}_2}$$

$$\forall x, y.\, x \in V_{\text{Expr}_1} \leftarrow (x = (2,4) + y \wedge y \in V_{\text{Expr}_1}) \vee x = (1,1)$$

$$\forall x., y\; x \in V_{\text{Expr}_2} \leftarrow (x = (3,6) + y \wedge y \in V_{\text{Expr}_1}) \vee (0,0) \in V_{\text{Expr}_2}$$

assert $(5,6) \in V_{\text{Start}}$

- Solvable by off-the-shelf Constrained Horn Clauses (CHC) solver
- Complete, but undecidable

Iterative approach: Solving Equations with Semi-linear Sets

- All variables and constants are semi-linear sets
- Operators: $\cup$, $+$, ITE

Equations

$$V_{\text{Start}} = V_{\text{Expr}_1} \cup V_{\text{Expr}_2}$$

$$V_{\text{Expr}_1} = \{(2,4)\} + V_{\text{Expr}_1} \cup \{(1,1)\}$$

$$V_{\text{Expr}_2} = \{(3,6)\} + V_{\text{Expr}_2} \cup \{(0,0)\}$$

?

Solution

$$V_{\text{Start}} = \{(1,1) + \lambda(2,4)\} \cup \{(0,0) + \lambda(3,6)\}$$

$$V_{\text{Expr}_1} = \{(1,1) + \lambda(2,4)\}$$

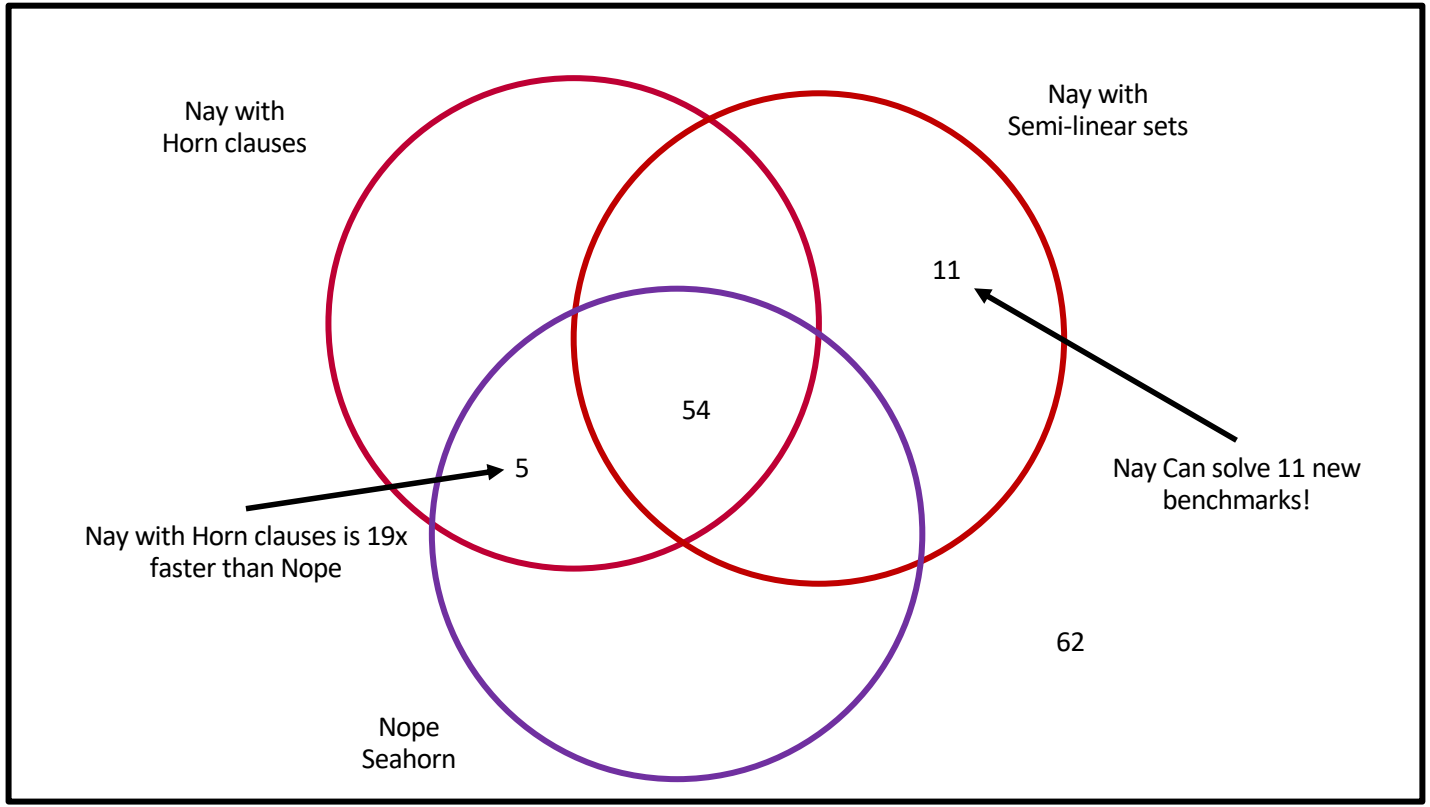
$$V_{\text{Expr}_2} = \{(0,0) + \lambda(3,6)\}$$

- Fact: These equations can be solved iteratively in no more than n rounds with n number of equations [Esparza et al. 2010]

Theorem

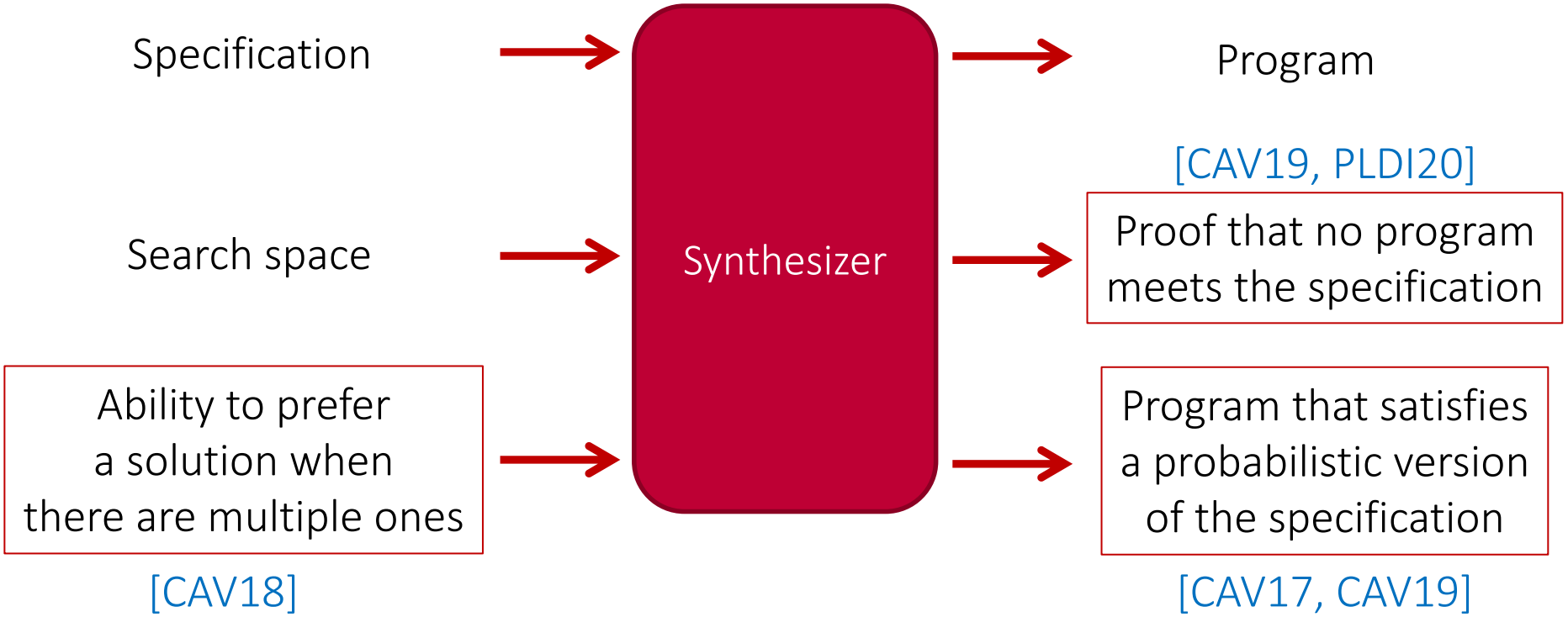
Given a conditional linear integer arithmetic (CLIA) SyGuS problem sy and a finite set of examples E, it is decidable whether the SyGuS problem sy^E is (un)realizable

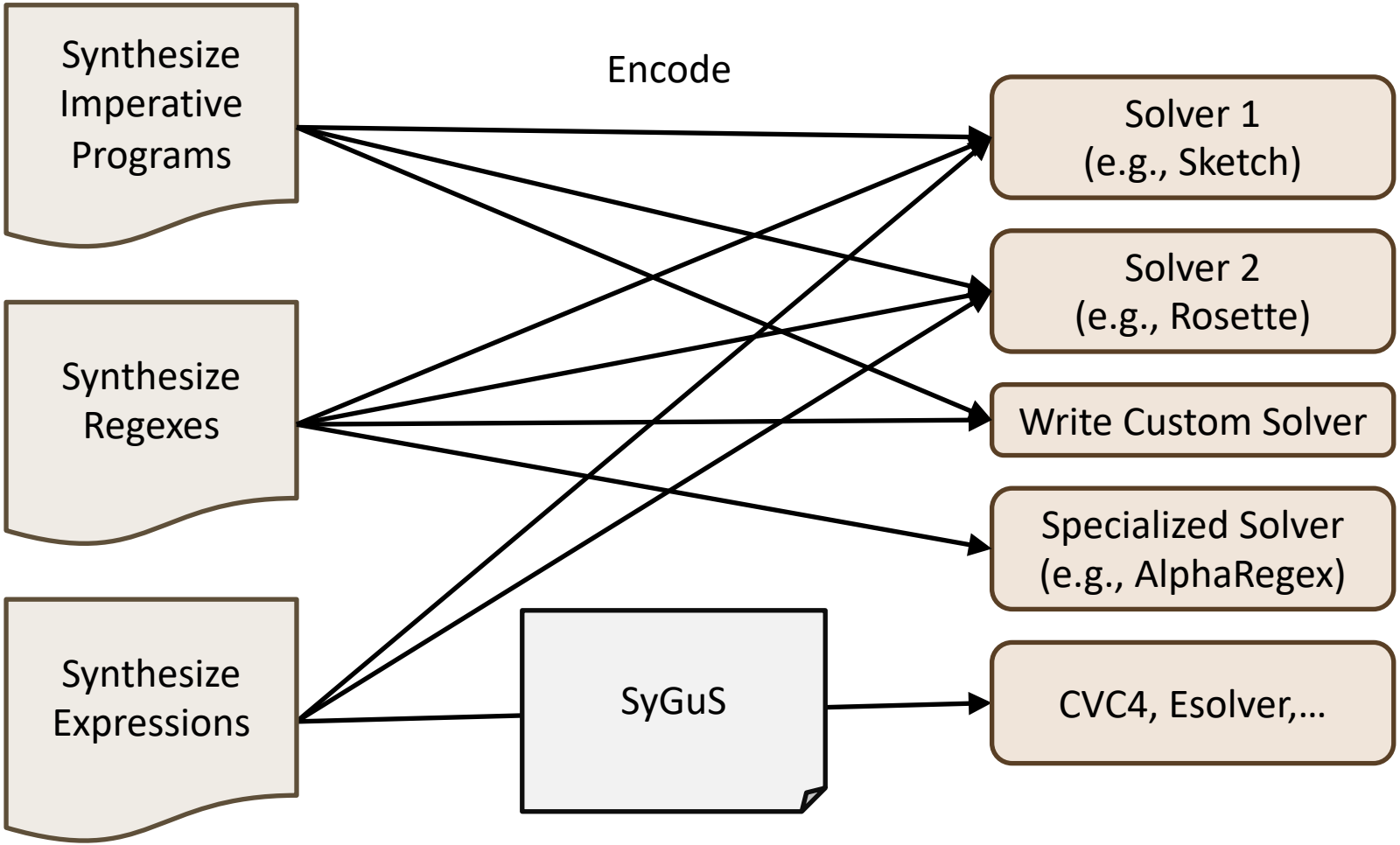
Nay (equation-based) vs Nope (reachability-based)

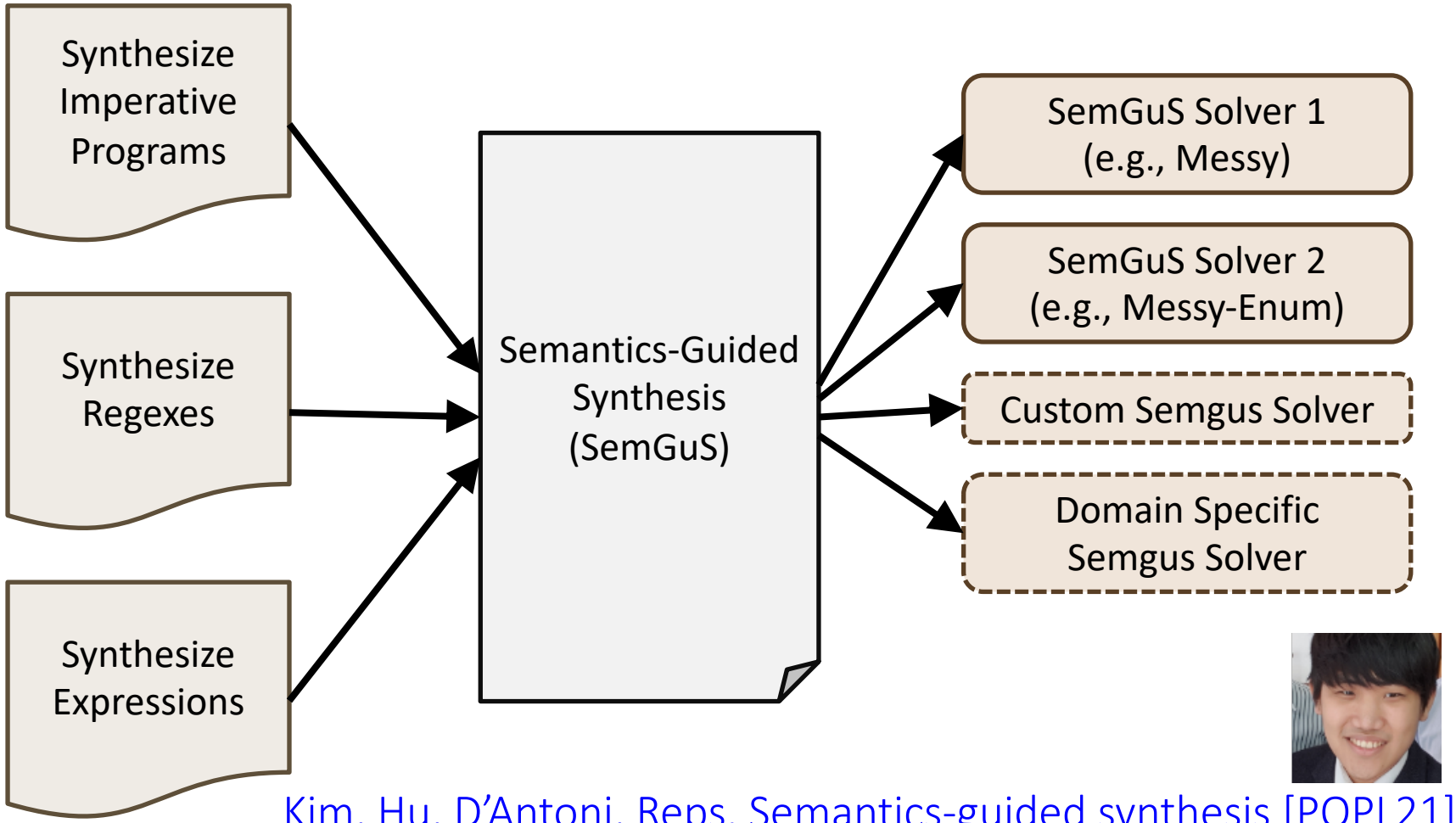


CONCLUSION

Program synthesis with guarantees




Synthesize Imperative Programs
Synthesize Regexes
Synthesize Expressions
Encode
SyGuS
Solver 1 (e.g., Sketch)
Solver 2 (e.g., Rosette)
Write Custom Solver
Specialized Solver (e.g., AlphaRegex)
CVC4, Esolver,…



Kim, Hu, D'Antoni, Reps. Semantics-guided synthesis [POPL21]