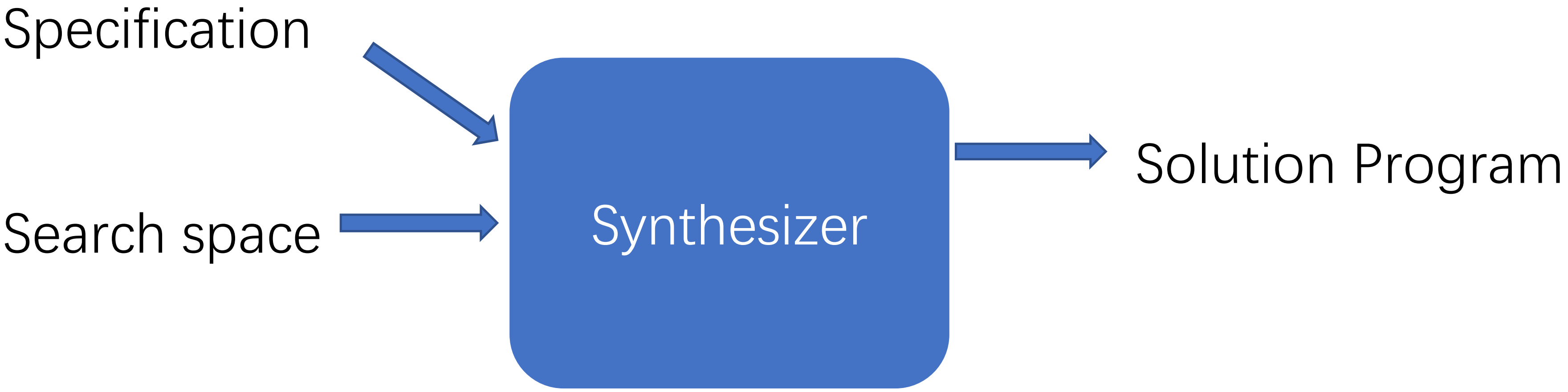
Guarantees in Program Synthesis

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

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

madPL

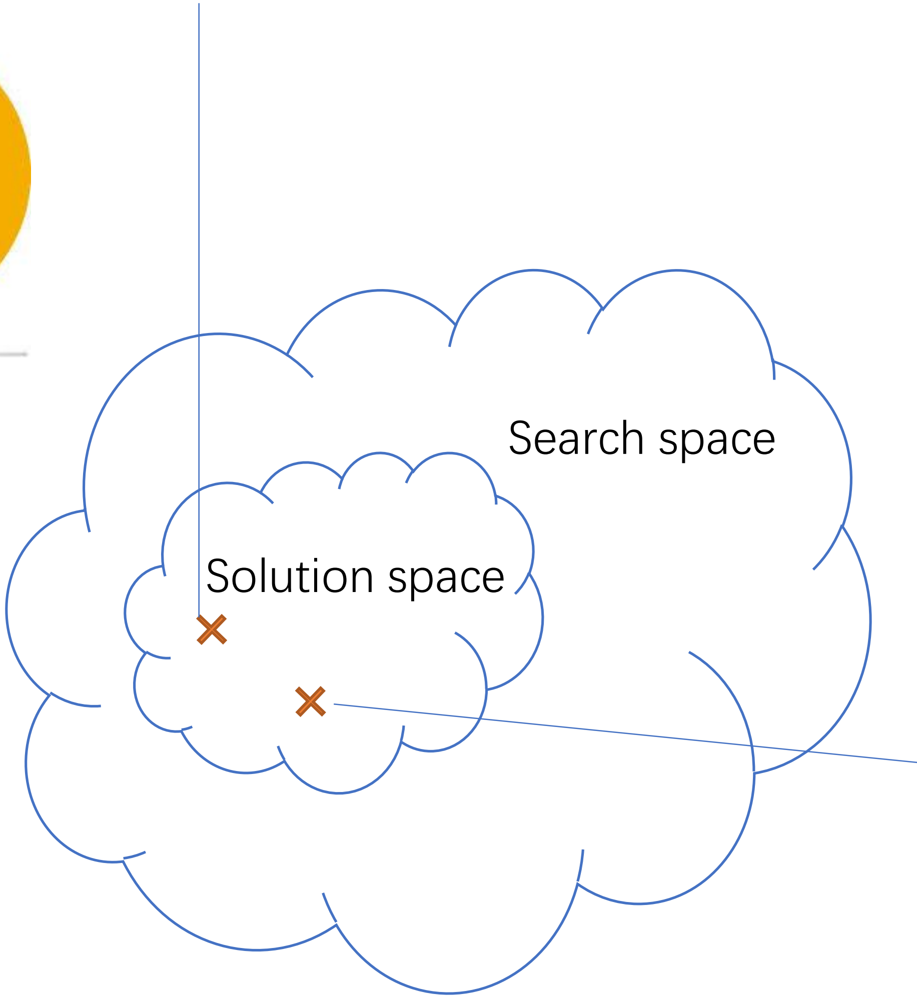
Program Synthesis

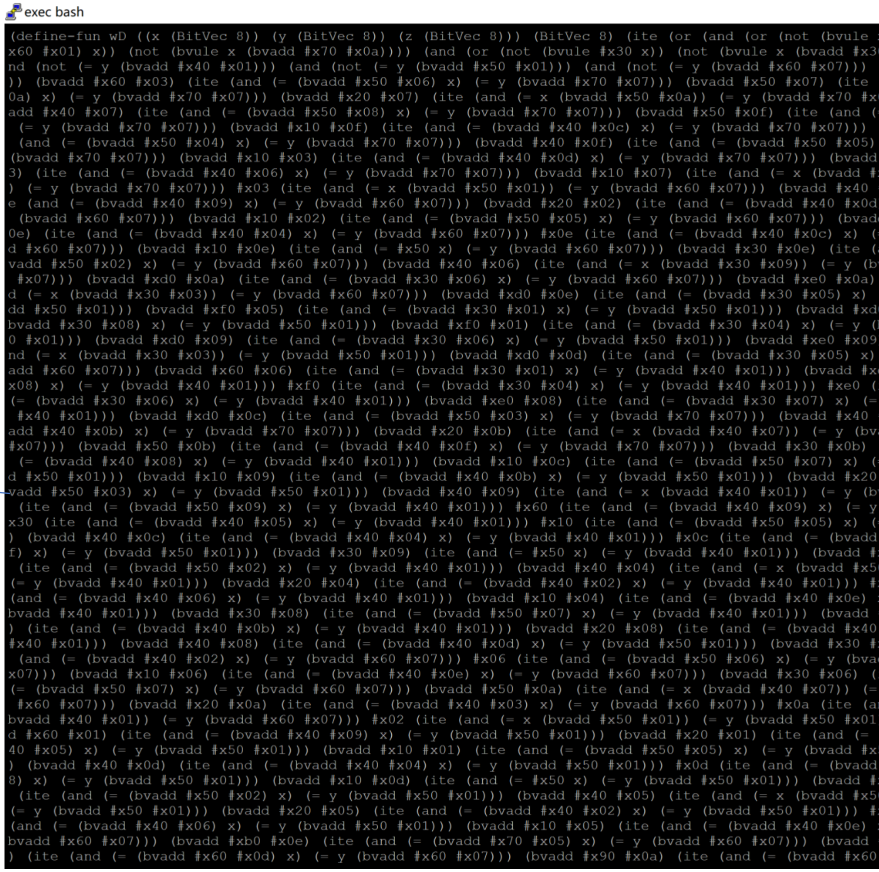


Program Synthesis is **Unpredictable**

Program Synthesis is **Unpredictable**

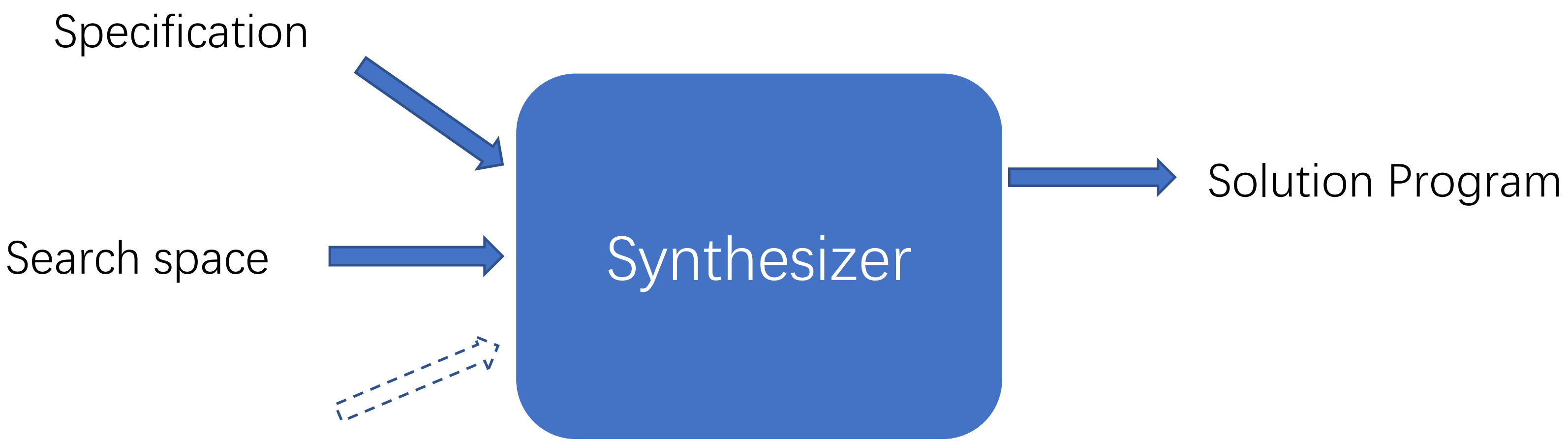
```
(define-fun ((x (BitVec 8)) (y (BitVec 8))) (bvand (bvlshl (DD x) #x02) (bvlshr (DD y) #x06)))
```





```
(define-fun wD ((x (BitVec 8)) (y (BitVec 8)) (z (BitVec 8))) (BitVec 8) (ite (or (and (or (not (bvule
x60 #x01) x)) (not (bvule x (bvadd #x70 #x0a)))) (and (or (not (bvule #x30 x)) (not (bvule x (bvadd #x3
nd (not (= y (bvadd #x40 #x01))) (and (not (= y (bvadd #x50 #x01))) (and (not (= y (bvadd #x60 #x07)))
)) (bvadd #x60 #x03) (ite (and (= (bvadd #x50 #x06) x) (= y (bvadd #x70 #x07))) (bvadd #x50 #x07) (ite
0a) x) (= y (bvadd #x70 #x07))) (bvadd #x20 #x07) (ite (and (= x (bvadd #x50 #x0a)) (= y (bvadd #x70 #x
add #x40 #x07) (ite (and (= (bvadd #x50 #x08) x) (= y (bvadd #x70 #x07))) (bvadd #x50 #x0f) (ite (and (
 (= y (bvadd #x70 #x07))) (bvadd #x10 #x0f) (ite (and (= (bvadd #x40 #x0c) x) (= y (bvadd #x70 #x07)))
 (and (= (bvadd #x50 #x04) x) (= y (bvadd #x70 #x07))) (bvadd #x40 #x0f) (ite (and (= (bvadd #x50 #x05)
(bvadd #x70 #x07))) (bvadd #x10 #x03) (ite (and (= (bvadd #x40 #x0d) x) (= y (bvadd #x70 #x07))) (bvadd
3) (ite (and (= (bvadd #x40 #x06) x) (= y (bvadd #x70 #x07))) (bvadd #x10 #x07) (ite (and (= x (bvadd #
) (= y (bvadd #x70 #x07))) #x03 (ite (and (= x (bvadd #x50 #x01)) (= y (bvadd #x60 #x07))) (bvadd #x40
e (and (= (bvadd #x40 #x09) x) (= y (bvadd #x60 #x07))) (bvadd #x20 #x02) (ite (and (= (bvadd #x40 #x0d
 (bvadd #x60 #x07))) (bvadd #x10 #x02) (ite (and (= (bvadd #x50 #x05) x) (= y (bvadd #x60 #x07))) (bvad
0e) (ite (and (= (bvadd #x40 #x04) x) (= y (bvadd #x60 #x07))) #x0e (ite (and (= (bvadd #x40 #x0c) x) (
d #x60 #x07))) (bvadd #x10 #x0e) (ite (and (= #x50 x) (= y (bvadd #x60 #x07))) (bvadd #x30 #x0e) (ite (
vadd #x50 #x02) x) (= y (bvadd #x60 #x07))) (bvadd #x40 #x06) (ite (and (= x (bvadd #x30 #x09)) (= y (b
 #x07))) (bvadd #xd0 #x0a) (ite (and (= (bvadd #x30 #x06) x) (= y (bvadd #x60 #x07))) (bvadd #xe0 #x0a)
d (= x (bvadd #x30 #x03)) (= y (bvadd #x60 #x07))) (bvadd #xd0 #x0e) (ite (and (= (bvadd #x30 #x05) x)
dd #x50 #x01))) (bvadd #xf0 #x05) (ite (and (= (bvadd #x30 #x01) x) (= y (bvadd #x50 #x01))) (bvadd #xd
bvadd #x30 #x08) x) (= y (bvadd #x50 #x01))) (bvadd #xf0 #x01) (ite (and (= (bvadd #x30 #x04) x) (= y (
0 #x01))) (bvadd #xd0 #x09) (ite (and (= (bvadd #x30 #x06) x) (= y (bvadd #x50 #x01))) (bvadd #xe0 #x09
nd (= x (bvadd #x30 #x03)) (= y (bvadd #x50 #x01))) (bvadd #xd0 #x0d) (ite (and (= (bvadd #x30 #x05) x)
add #x60 #x07))) (bvadd #x60 #x06) (ite (and (= (bvadd #x30 #x01) x) (= y (bvadd #x40 #x01))) (bvadd #x
x08) x) (= y (bvadd #x40 #x01))) #xf0 (ite (and (= (bvadd #x30 #x04) x) (= y (bvadd #x40 #x01))) #xe0 (
(= (bvadd #x30 #x06) x) (= y (bvadd #x40 #x01))) (bvadd #xe0 #x08) (ite (and (= (bvadd #x30 #x07) x) (=
 #x40 #x01))) (bvadd #xd0 #x0c) (ite (and (= (bvadd #x50 #x03) x) (= y (bvadd #x70 #x07))) (bvadd #x40
add #x40 #x0b) x) (= y (bvadd #x70 #x07))) (bvadd #x20 #x0b) (ite (and (= x (bvadd #x40 #x07)) (= y (bv
#x07))) (bvadd #x50 #x0b) (ite (and (= (bvadd #x40 #x0f) x) (= y (bvadd #x70 #x07))) (bvadd #x30 #x0b)
 (= (bvadd #x40 #x08) x) (= y (bvadd #x40 #x01))) (bvadd #x10 #x0c) (ite (and (= (bvadd #x50 #x07) x) (
d #x50 #x01))) (bvadd #x10 #x09) (ite (and (= (bvadd #x40 #x0b) x) (= y (bvadd #x50 #x01))) (bvadd #x20
vadd #x50 #x03) x) (= y (bvadd #x50 #x01))) (bvadd #x40 #x09) (ite (and (= x (bvadd #x40 #x01)) (= y (b
 (ite (and (= (bvadd #x50 #x09) x) (= y (bvadd #x40 #x01))) #x60 (ite (and (= (bvadd #x40 #x09) x) (= y
x30 (ite (and (= (bvadd #x40 #x05) x) (= y (bvadd #x40 #x01))) #x10 (ite (and (= (bvadd #x50 #x05) x) (
) (bvadd #x40 #x0c) (ite (and (= (bvadd #x40 #x04) x) (= y (bvadd #x40 #x01))) #x0c (ite (and (= (bvadd
f) x) (= y (bvadd #x50 #x01))) (bvadd #x30 #x09) (ite (and (= #x50 x) (= y (bvadd #x40 #x01))) (bvadd #
 (ite (and (= (bvadd #x50 #x02) x) (= y (bvadd #x40 #x01))) (bvadd #x40 #x04) (ite (and (= x (bvadd #x5
(= y (bvadd #x40 #x01))) (bvadd #x20 #x04) (ite (and (= (bvadd #x40 #x02) x) (= y (bvadd #x40 #x01))) #
(and (= (bvadd #x40 #x06) x) (= y (bvadd #x40 #x01))) (bvadd #x10 #x04) (ite (and (= (bvadd #x40 #x0e)
bvadd #x40 #x01))) (bvadd #x30 #x08) (ite (and (= (bvadd #x50 #x07) x) (= y (bvadd #x40 #x01))) (bvadd
) (ite (and (= (bvadd #x40 #x0b) x) (= y (bvadd #x40 #x01))) (bvadd #x20 #x08) (ite (and (= (bvadd #x40
#x40 #x01))) (bvadd #x40 #x08) (ite (and (= (bvadd #x40 #x0d) x) (= y (bvadd #x50 #x01))) (bvadd #x30 #
 (and (= (bvadd #x40 #x02) x) (= y (bvadd #x60 #x07))) #x06 (ite (and (= (bvadd #x50 #x06) x) (= y (bva
x07))) (bvadd #x10 #x06) (ite (and (= (bvadd #x40 #x0e) x) (= y (bvadd #x60 #x07))) (bvadd #x30 #x06) (
(= (bvadd #x50 #x07) x) (= y (bvadd #x60 #x07))) (bvadd #x50 #x0a) (ite (and (= x (bvadd #x40 #x07)) (=
 #x60 #x07))) (bvadd #x20 #x0a) (ite (and (= (bvadd #x40 #x03) x) (= y (bvadd #x60 #x07))) #x0a (ite (a
bvadd #x40 #x01)) (= y (bvadd #x60 #x07))) #x02 (ite (and (= x (bvadd #x50 #x01)) (= y (bvadd #x50 #x01
d #x60 #x01) (ite (and (= (bvadd #x40 #x09) x) (= y (bvadd #x50 #x01))) (bvadd #x20 #x01) (ite (and (=
40 #x05) x) (= y (bvadd #x50 #x01))) (bvadd #x10 #x01) (ite (and (= (bvadd #x50 #x05) x) (= y (bvadd #x
) (bvadd #x40 #x0d) (ite (and (= (bvadd #x40 #x04) x) (= y (bvadd #x50 #x01))) #x0d (ite (and (= (bvadd
8) x) (= y (bvadd #x50 #x01))) (bvadd #x10 #x0d) (ite (and (= #x50 x) (= y (bvadd #x50 #x01))) (bvadd #
 (ite (and (= (bvadd #x50 #x02) x) (= y (bvadd #x50 #x01))) (bvadd #x40 #x05) (ite (and (= x (bvadd #x5
(= y (bvadd #x50 #x01))) (bvadd #x20 #x05) (ite (and (= (bvadd #x40 #x02) x) (= y (bvadd #x50 #x01))) #
(and (= (bvadd #x40 #x06) x) (= y (bvadd #x50 #x01))) (bvadd #x10 #x05) (ite (and (= (bvadd #x40 #x0e)
bvadd #x60 #x07))) (bvadd #xb0 #x0e) (ite (and (= (bvadd #x70 #x05) x) (= y (bvadd #x60 #x07))) (bvadd
) (ite (and (= (bvadd #x60 #x0d) x) (= y (bvadd #x60 #x07))) (bvadd #x90 #x0a) (ite (and (= (bvadd #x60
```

Program Synthesis is **Unpredictable**



Specification

Search space

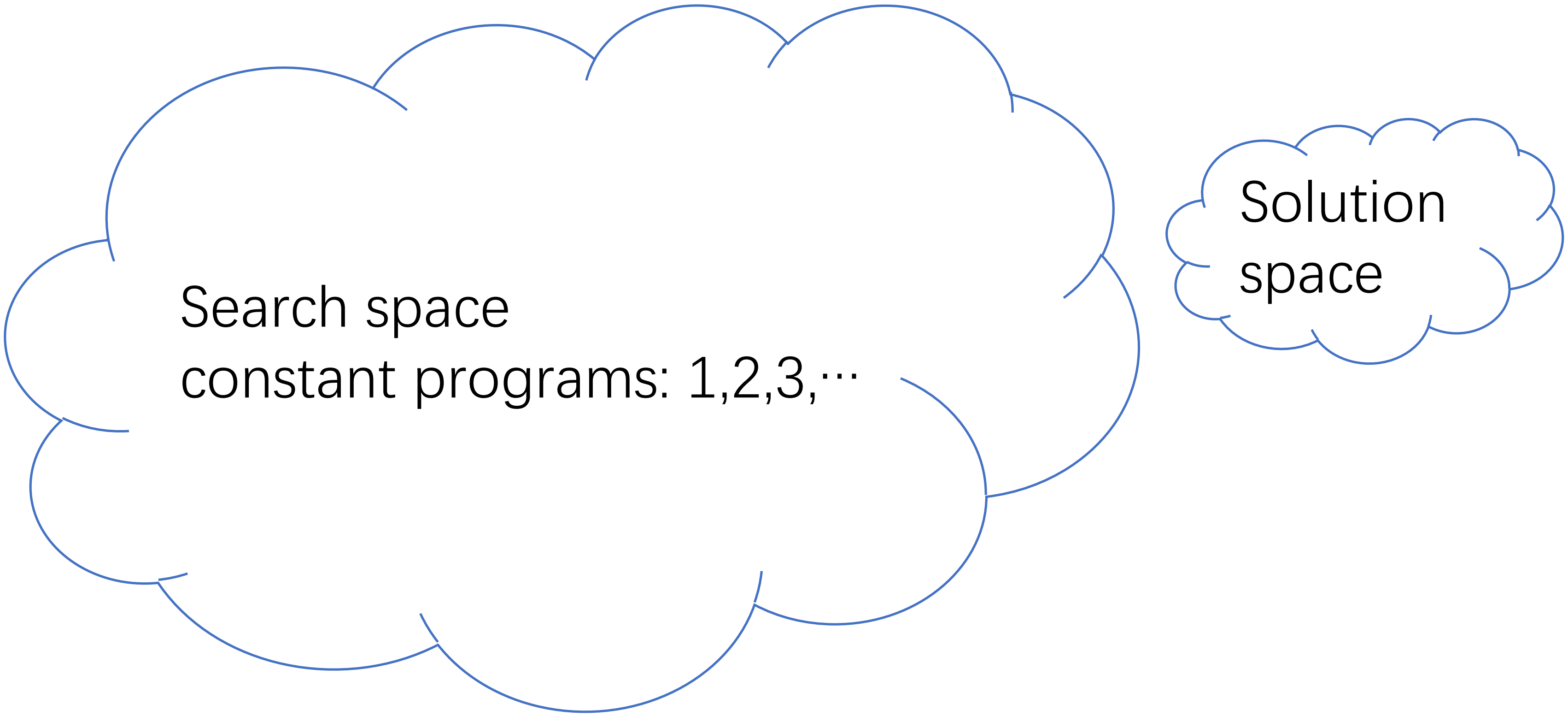
Synthesizer

Solution Program

Ability to prefer a solution
when there are multiple solution

Program Synthesis is **Unpredictable**

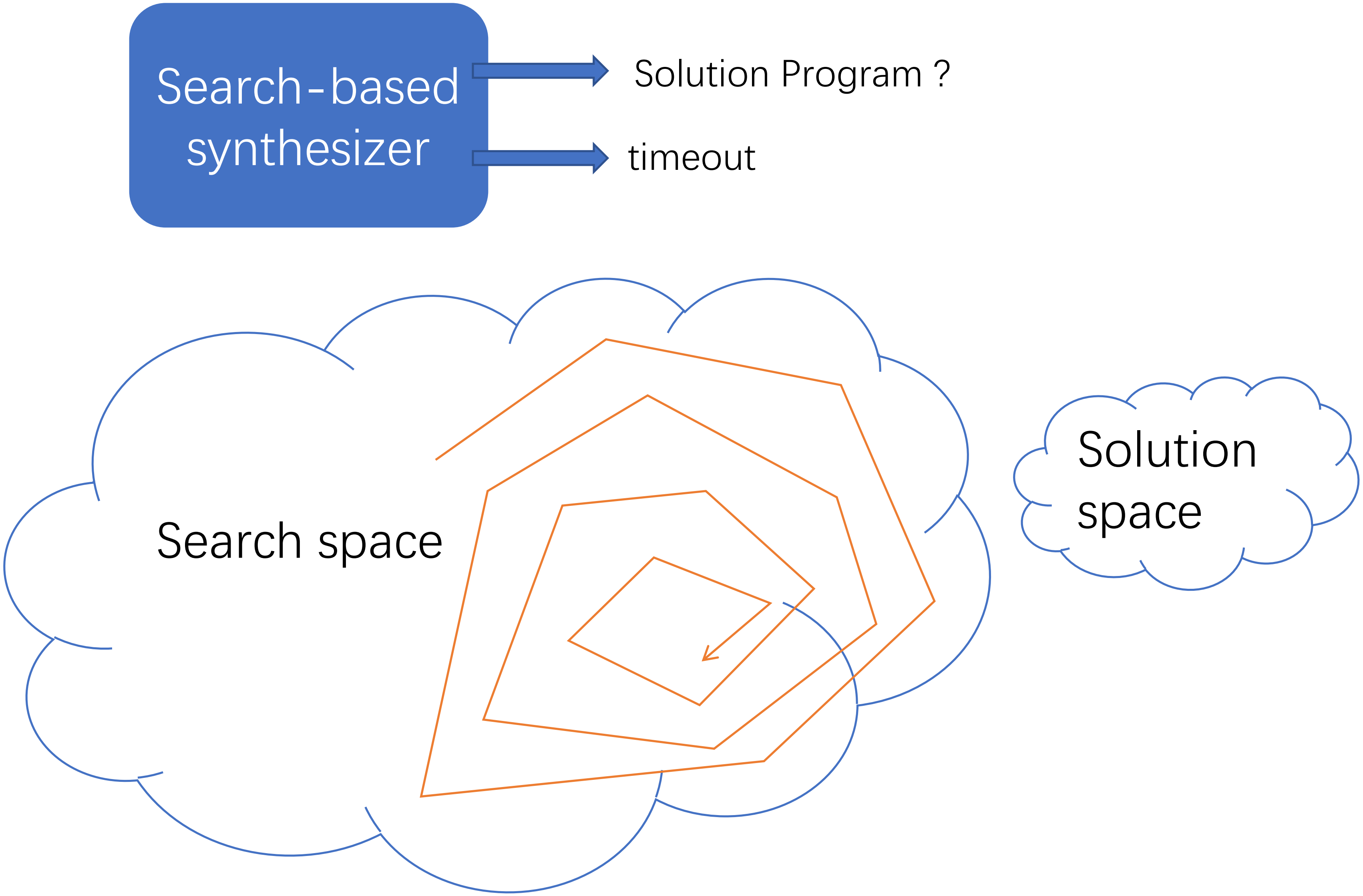
Specification: $f(x) = x$



Search space
constant programs: 1,2,3,⋯

Solution space

Program Synthesis is **Unpredictable**



Search-based synthesizer

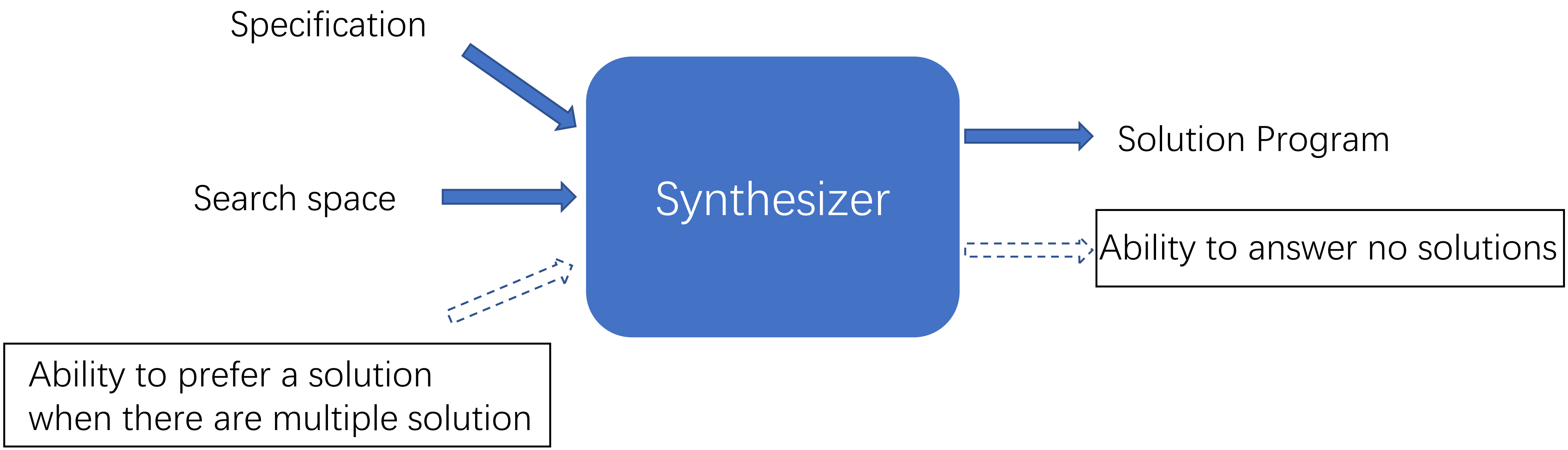
Solution Program ?

timeout

Search space

Solution space

Guarantees in Program Synthesis



Specification

Search space

Synthesizer

Solution Program

Ability to answer no solutions

Ability to prefer a solution when there are multiple solution

Make program synthesis predictable

Syntax-Guided Synthesis with Quantitative Objectives [CAV18]

Syntax-Guided Synthesis (SyGuS)

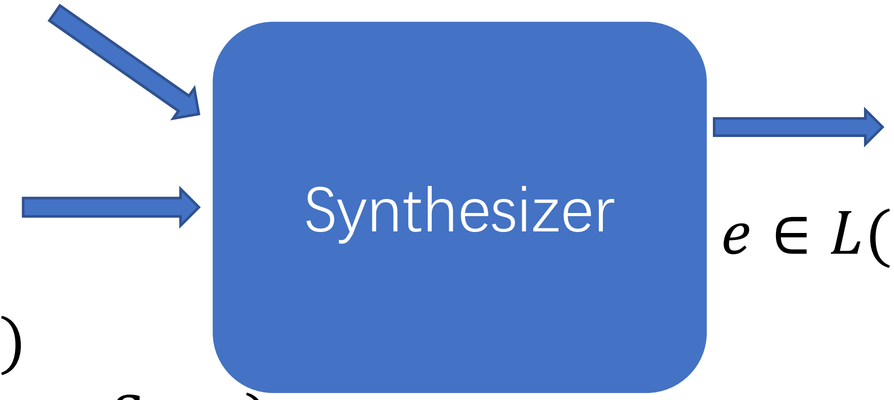
$\varphi(max(x,y),x,y): max(x,y) \geq x \wedge max(x,y) \geq y \wedge (max(x,y) = x \vee max(x,y) = y)$

Specification

Search space

$$
\begin{aligned}
\text{Start} = &+(\text{Start}, \text{Start}) \\
&\mid ITE(\text{BExpr}, \text{Start}, \text{Start}) \\
&\mid x \mid y \mid 0 \mid 1
\end{aligned}
$$

$$
\begin{aligned}
\text{BExpr} = &\ Not(\text{BExpr}) \\
&\mid > (\text{Start}, \text{Start}) \\
&\mid And(\text{BExpr}, \text{BExpr})
\end{aligned}
$$



Solution Program

$e \in L(G)$ such that

$\forall x, y. \varphi(e, x, y)$

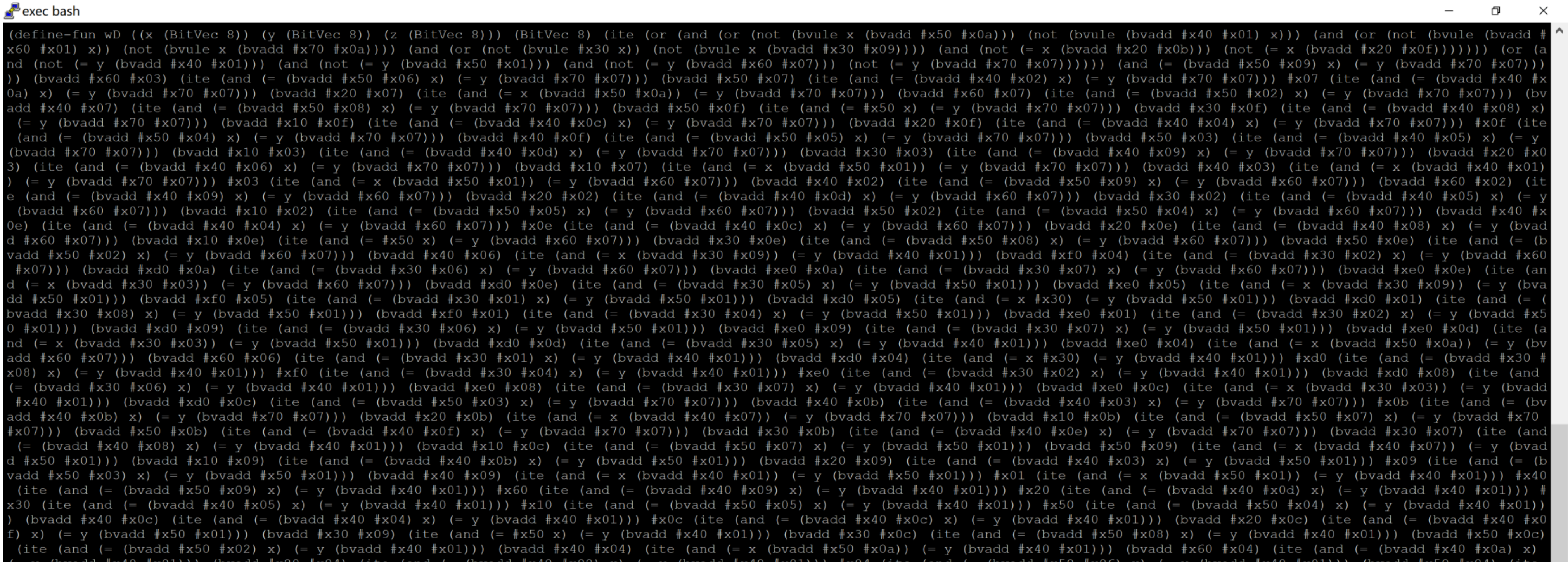
$\boxed{\max(x,y) = ITE(> (x,y), x, ITE(< (x,y), y, x)}$

A limitation of SyGuS

What we expected

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

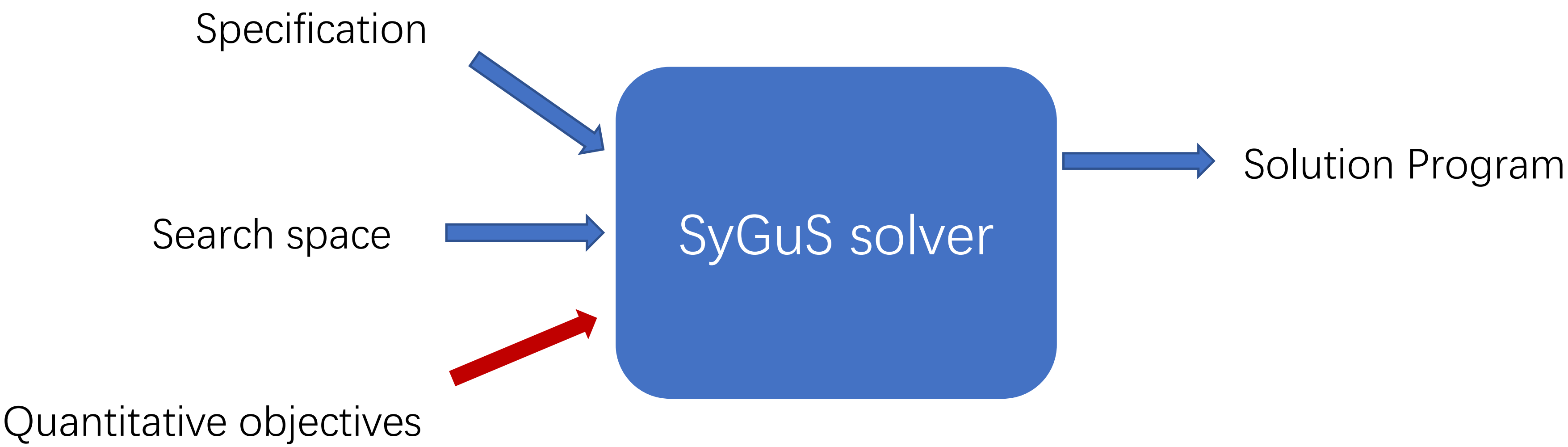
Output of the CVC4 solver



```
(define-fun wD ((x (BitVec 8)) (y (BitVec 8)) (z (BitVec 8))) (BitVec 8) (ite (or (and (or (not (bvule x (bvadd #x50 #x0a))) (not (bvule (bvadd #x40 #x01) x))) (and (or (not (bvule (bvadd #
x60 #x01) x)) (not (bvule x (bvadd #x70 #x0a)))) (and (or (not (bvule #x30 x)) (not (bvule x (bvadd #x30 #x09)))) (and (not (= x (bvadd #x20 #x0b))) (not (= x (bvadd #x20 #x0f))))))) (or (a
nd (not (= y (bvadd #x40 #x01))) (and (not (= y (bvadd #x50 #x01))) (and (not (= y (bvadd #x60 #x07))) (not (= y (bvadd #x70 #x07)))))) (and (= (bvadd #x50 #x09) x) (= y (bvadd #x70 #x07)))
)) (bvadd #x60 #x03) (ite (and (= (bvadd #x50 #x06) x) (= y (bvadd #x70 #x07))) (bvadd #x50 #x07) (ite (and (= (bvadd #x40 #x02) x) (= y (bvadd #x70 #x07))) #x07 (ite (and (= (bvadd #x40 #x
0a) x) (= y (bvadd #x70 #x07))) (bvadd #x20 #x07) (ite (and (= x (bvadd #x50 #x0a)) (= y (bvadd #x70 #x07))) (bvadd #x60 #x07) (ite (and (= (bvadd #x50 #x02) x) (= y (bvadd #x70 #x07))) (bv
add #x40 #x07) (ite (and (= (bvadd #x50 #x08) x) (= y (bvadd #x70 #x07))) (bvadd #x50 #x0f) (ite (and (= #x50 x) (= y (bvadd #x70 #x07))) (bvadd #x30 #x0f) (ite (and (= (bvadd #x40 #x08) x)
 (= y (bvadd #x70 #x07))) (bvadd #x10 #x0f) (ite (and (= (bvadd #x40 #x0c) x) (= y (bvadd #x70 #x07))) (bvadd #x20 #x0f) (ite (and (= (bvadd #x40 #x04) x) (= y (bvadd #x70 #x07))) #x0f (ite
 (and (= (bvadd #x50 #x04) x) (= y (bvadd #x70 #x07))) (bvadd #x40 #x0f) (ite (and (= (bvadd #x50 #x05) x) (= y (bvadd #x70 #x07))) (bvadd #x50 #x03) (ite (and (= (bvadd #x40 #x05) x) (= y
(bvadd #x70 #x07))) (bvadd #x10 #x03) (ite (and (= (bvadd #x40 #x0d) x) (= y (bvadd #x70 #x07))) (bvadd #x30 #x03) (ite (and (= (bvadd #x40 #x09) x) (= y (bvadd #x70 #x07))) (bvadd #x20 #x0
3) (ite (and (= (bvadd #x40 #x06) x) (= y (bvadd #x70 #x07))) (bvadd #x10 #x07) (ite (and (= x (bvadd #x50 #x01)) (= y (bvadd #x70 #x07))) (bvadd #x40 #x03) (ite (and (= x (bvadd #x40 #x01)
) (= y (bvadd #x70 #x07))) #x03 (ite (and (= x (bvadd #x50 #x01)) (= y (bvadd #x60 #x07))) (bvadd #x40 #x02) (ite (and (= (bvadd #x50 #x09) x) (= y (bvadd #x60 #x07))) (bvadd #x60 #x02) (it
e (and (= (bvadd #x40 #x09) x) (= y (bvadd #x60 #x07))) (bvadd #x20 #x02) (ite (and (= (bvadd #x40 #x0d) x) (= y (bvadd #x60 #x07))) (bvadd #x30 #x02) (ite (and (= (bvadd #x40 #x05) x) (= y
 (bvadd #x60 #x07))) (bvadd #x10 #x02) (ite (and (= (bvadd #x50 #x05) x) (= y (bvadd #x60 #x07))) (bvadd #x50 #x02) (ite (and (= (bvadd #x50 #x04) x) (= y (bvadd #x60 #x07))) (bvadd #x40 #x
0e) (ite (and (= (bvadd #x40 #x04) x) (= y (bvadd #x60 #x07))) #x0e (ite (and (= (bvadd #x40 #x0c) x) (= y (bvadd #x60 #x07))) (bvadd #x20 #x0e) (ite (and (= (bvadd #x40 #x08) x) (= y (bvad
d #x60 #x07))) (bvadd #x10 #x0e) (ite (and (= #x50 x) (= y (bvadd #x60 #x07))) (bvadd #x30 #x0e) (ite (and (= (bvadd #x50 #x08) x) (= y (bvadd #x60 #x07))) (bvadd #x50 #x0e) (ite (and (= (b
vadd #x50 #x02) x) (= y (bvadd #x60 #x07))) (bvadd #x40 #x06) (ite (and (= x (bvadd #x30 #x09)) (= y (bvadd #x40 #x01))) (bvadd #xf0 #x04) (ite (and (= (bvadd #x30 #x02) x) (= y (bvadd #x60
 #x07))) (bvadd #xd0 #x0a) (ite (and (= (bvadd #x30 #x06) x) (= y (bvadd #x60 #x07))) (bvadd #xe0 #x0a) (ite (and (= (bvadd #x30 #x07) x) (= y (bvadd #x60 #x07))) (bvadd #xe0 #x0e) (ite (an
d (= x (bvadd #x30 #x03)) (= y (bvadd #x60 #x07))) (bvadd #xd0 #x0e) (ite (and (= (bvadd #x30 #x05) x) (= y (bvadd #x50 #x01))) (bvadd #xe0 #x05) (ite (and (= x (bvadd #x30 #x09)) (= y (bva
dd #x50 #x01))) (bvadd #xf0 #x05) (ite (and (= (bvadd #x30 #x01) x) (= y (bvadd #x50 #x01))) (bvadd #xd0 #x05) (ite (and (= x #x30) (= y (bvadd #x50 #x01))) (bvadd #xd0 #x01) (ite (and (= (
bvadd #x30 #x08) x) (= y (bvadd #x50 #x01))) (bvadd #xf0 #x01) (ite (and (= (bvadd #x30 #x04) x) (= y (bvadd #x50 #x01))) (bvadd #xe0 #x01) (ite (and (= (bvadd #x30 #x02) x) (= y (bvadd #x5
0 #x01))) (bvadd #xd0 #x09) (ite (and (= (bvadd #x30 #x06) x) (= y (bvadd #x50 #x01))) (bvadd #xe0 #x09) (ite (and (= (bvadd #x30 #x07) x) (= y (bvadd #x50 #x01))) (bvadd #xe0 #x0d) (ite (a
nd (= x (bvadd #x30 #x03)) (= y (bvadd #x50 #x01))) (bvadd #xd0 #x0d) (ite (and (= (bvadd #x30 #x05) x) (= y (bvadd #x40 #x01))) (bvadd #xe0 #x04) (ite (and (= x (bvadd #x50 #x0a)) (= y (bv
add #x60 #x07))) (bvadd #x60 #x06) (ite (and (= (bvadd #x30 #x01) x) (= y (bvadd #x40 #x01))) (bvadd #xd0 #x04) (ite (and (= x #x30) (= y (bvadd #x40 #x01))) #xd0 (ite (and (= (bvadd #x30 #
x08) x) (= y (bvadd #x40 #x01))) #xf0 (ite (and (= (bvadd #x30 #x04) x) (= y (bvadd #x40 #x01))) #xe0 (ite (and (= (bvadd #x30 #x02) x) (= y (bvadd #x40 #x01))) (bvadd #xd0 #x08) (ite (and
(= (bvadd #x30 #x06) x) (= y (bvadd #x40 #x01))) (bvadd #xe0 #x08) (ite (and (= (bvadd #x30 #x07) x) (= y (bvadd #x40 #x01))) (bvadd #xe0 #x0c) (ite (and (= x (bvadd #x30 #x03)) (= y (bvadd
 #x40 #x01))) (bvadd #xd0 #x0c) (ite (and (= (bvadd #x50 #x03) x) (= y (bvadd #x70 #x07))) (bvadd #x40 #x0b) (ite (and (= (bvadd #x40 #x03) x) (= y (bvadd #x70 #x07))) #x0b (ite (and (= (bv
add #x40 #x0b) x) (= y (bvadd #x70 #x07))) (bvadd #x20 #x0b) (ite (and (= x (bvadd #x40 #x07)) (= y (bvadd #x70 #x07))) (bvadd #x10 #x0b) (ite (and (= (bvadd #x50 #x07) x) (= y (bvadd #x70
#x07))) (bvadd #x50 #x0b) (ite (and (= (bvadd #x40 #x0f) x) (= y (bvadd #x70 #x07))) (bvadd #x30 #x0b) (ite (and (= (bvadd #x40 #x0e) x) (= y (bvadd #x70 #x07))) (bvadd #x30 #x07) (ite (and
 (= (bvadd #x40 #x08) x) (= y (bvadd #x40 #x01))) (bvadd #x10 #x0c) (ite (and (= (bvadd #x50 #x07) x) (= y (bvadd #x50 #x01))) (bvadd #x50 #x09) (ite (and (= x (bvadd #x40 #x07)) (= y (bvad
d #x50 #x01))) (bvadd #x10 #x09) (ite (and (= (bvadd #x40 #x0b) x) (= y (bvadd #x50 #x01))) (bvadd #x20 #x09) (ite (and (= (bvadd #x40 #x03) x) (= y (bvadd #x50 #x01))) #x09 (ite (and (= (b
vadd #x50 #x03) x) (= y (bvadd #x50 #x01))) (bvadd #x40 #x09) (ite (and (= x (bvadd #x40 #x01)) (= y (bvadd #x50 #x01))) #x01 (ite (and (= x (bvadd #x50 #x01)) (= y (bvadd #x40 #x01))) #x40
 (ite (and (= (bvadd #x50 #x09) x) (= y (bvadd #x40 #x01))) #x60 (ite (and (= (bvadd #x40 #x09) x) (= y (bvadd #x40 #x01))) #x20 (ite (and (= (bvadd #x40 #x0d) x) (= y (bvadd #x40 #x01))) #
x30 (ite (and (= (bvadd #x40 #x05) x) (= y (bvadd #x40 #x01))) #x10 (ite (and (= (bvadd #x50 #x05) x) (= y (bvadd #x40 #x01))) #x50 (ite (and (= (bvadd #x50 #x04) x) (= y (bvadd #x40 #x01))
) (bvadd #x40 #x0c) (ite (and (= (bvadd #x40 #x04) x) (= y (bvadd #x40 #x01))) #x0c (ite (and (= (bvadd #x40 #x0c) x) (= y (bvadd #x40 #x01))) (bvadd #x20 #x0c) (ite (and (= (bvadd #x40 #x0
f) x) (= y (bvadd #x50 #x01))) (bvadd #x30 #x09) (ite (and (= #x50 x) (= y (bvadd #x40 #x01))) (bvadd #x30 #x0c) (ite (and (= (bvadd #x50 #x08) x) (= y (bvadd #x40 #x01))) (bvadd #x50 #x0c)
 (ite (and (= (bvadd #x50 #x02) x) (= y (bvadd #x40 #x01))) (bvadd #x40 #x04) (ite (and (= x (bvadd #x50 #x0a)) (= y (bvadd #x40 #x01))) (bvadd #x60 #x04) (ite (and (= (bvadd #x40 #x0a) x)
```

Features you want	Optimization Objectives
Readable	Smallest size
Efficient	Least number of multiplication
Most likely	Largest probability

Guarantees in SyGuS



Specification

Search space

Quantitative objectives

SyGuS solver

Solution Program

Adding Quantitative Objective to SyGuS

$$\begin{aligned}\mathrm{Start} = {} & \mathrm{Start} + \mathrm{Start} \\ & \mid \mathit{if}\,(\mathrm{BExpr})\ \mathit{then}\ \mathrm{Start}\ \mathit{else}\ \mathrm{Start} \\ & \mid \mathrm{x} \quad \mid y \quad \mid 0 \quad \mid 1\end{aligned}$$

$$\begin{aligned}\mathrm{BExpr} = {} & \mathrm{Start} > \mathrm{Start} \\ & \mid \mathit{not}\ \mathrm{BExpr} \\ & \mid \mathrm{BExpr}\ \mathit{and}\ \mathrm{BExpr}\end{aligned}$$

Adding Quantitative Objective to SyGuS

$$\begin{aligned}\mathrm{Start} = {} & \mathrm{Start} + \mathrm{Start} \\ & \mid \mathit{if}\,(\mathrm{BExpr})\ \mathit{then}\ \mathrm{Start}\ \mathit{else}\ \mathrm{Start} \\ & \mid \mathrm{x} \quad \mid y \quad \mid 0 \quad \mid 1\end{aligned}$$

$$\begin{aligned}\mathrm{BExpr} = {} & \mathrm{Start} > \mathrm{Start} \\ & \mid \mathit{not}\ \mathrm{BExpr} \\ & \mid \mathrm{BExpr}\ \mathit{and}\ \mathrm{BExpr}\end{aligned}$$

Quantitative objective: Minimize number of if-statement

Adding Quantitative Objective to SyGuS

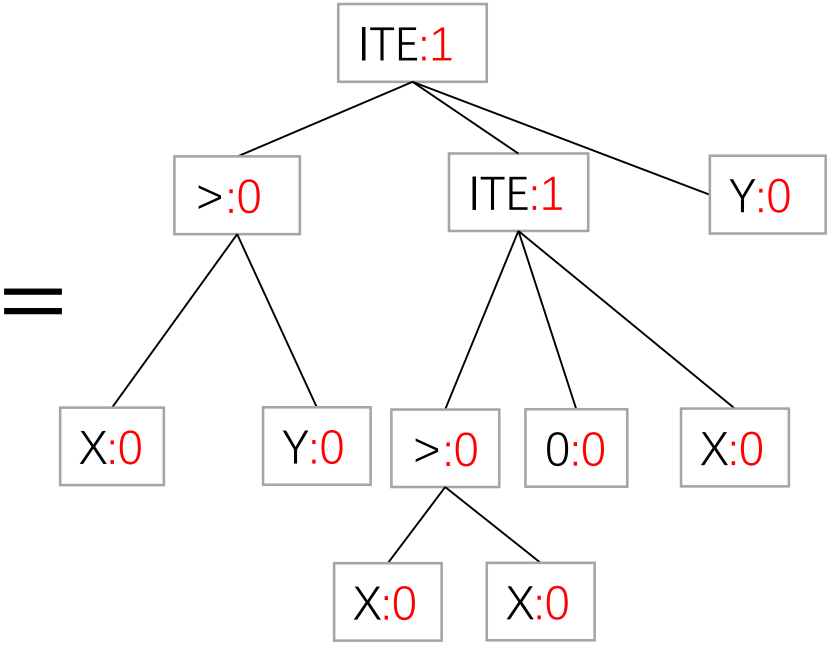
$$\begin{aligned}\mathrm{Start} = {} & \mathrm{Start} + \mathrm{Start}\ /0\\ & \mid \mathit{if}\,(\mathrm{BExpr})\ \mathit{then}\ \mathrm{Start}\ \mathit{else}\ \mathrm{Start}\ /1\\ & \mid \mathrm{x}/0 \mid y/0 \mid 0/0 \mid 1/0\end{aligned}$$

$$\begin{aligned}\mathrm{BExpr} = {} & \mathrm{Start} > \mathrm{Start}\ /0\\ & \mid \mathit{not}\ \mathrm{BExpr}\ /0\\ & \mid \mathrm{BExpr}\ \mathit{and}\ \mathrm{BExpr}\ /0\end{aligned}$$

Quantitative objective: Minimize number of if-statement

```
If(x>y)
        then
                if(x > x) then 0 else x
        Else y
```

=



Weight = 2

Adding Quantitative Objective to SyGuS

Weighted grammar

$$\begin{aligned} \text{Start} = {} & \text{Start} + \text{Start}\ /0 \\ & \mid if\,(\text{BExpr})\ then\ \text{Start}\ else\ \text{Start}\ /1 \\ & \mid \text{x}/0 \mid y/0 \mid 0/0 \mid 1/0 \end{aligned}$$

$$\begin{aligned} \text{BExpr} = {} & \text{Start} > \text{Start}\ /0 \\ & \mid not\ \text{BExpr}\ /0 \\ & \mid \text{BExpr}\ and\ \text{BExpr}\ /0 \end{aligned}$$

Quantitative objective: Minimize number of if-statement

Adding Quantitative Objective to SyGuS

Weighted grammar

$$\begin{aligned}\text{Start} = {} & \text{Start} + \text{Start}\ /0\\ & \mid if\,(\text{BExpr})\ then\ \text{Start}\ else\ \text{Start}\ /1\\ & \mid \text{x}/0 \mid y/0 \mid 0/0 \mid 1/0\end{aligned}$$

$$\begin{aligned}\text{BExpr} = {} & \text{Start} > \text{Start}\ /0\\ & \mid not\ \text{BExpr}\ /0\\ & \mid \text{BExpr}\ and\ \text{BExpr}\ /0\end{aligned}$$

Quantitative objective: Minimize number of if-statement

Minimize weight

QSyGuS: (specification,weighted grammar, Quantitative objective)

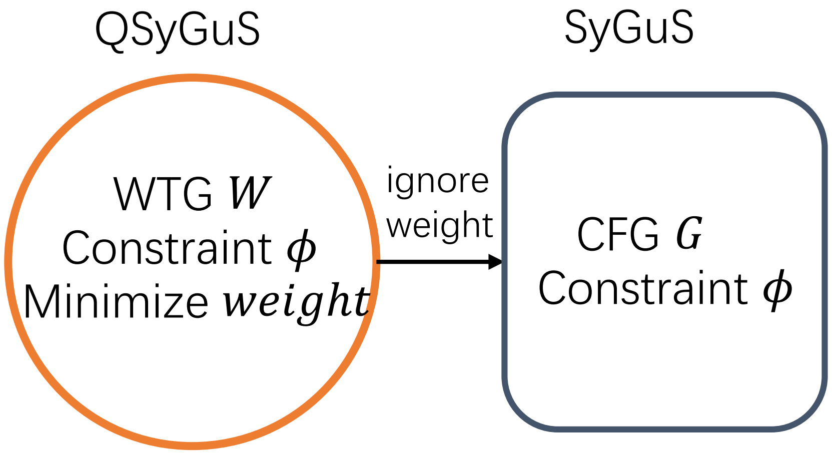
Solving QSyGuS

QSyGuS

WTG W

Constraint ϕ

Minimize $weight$



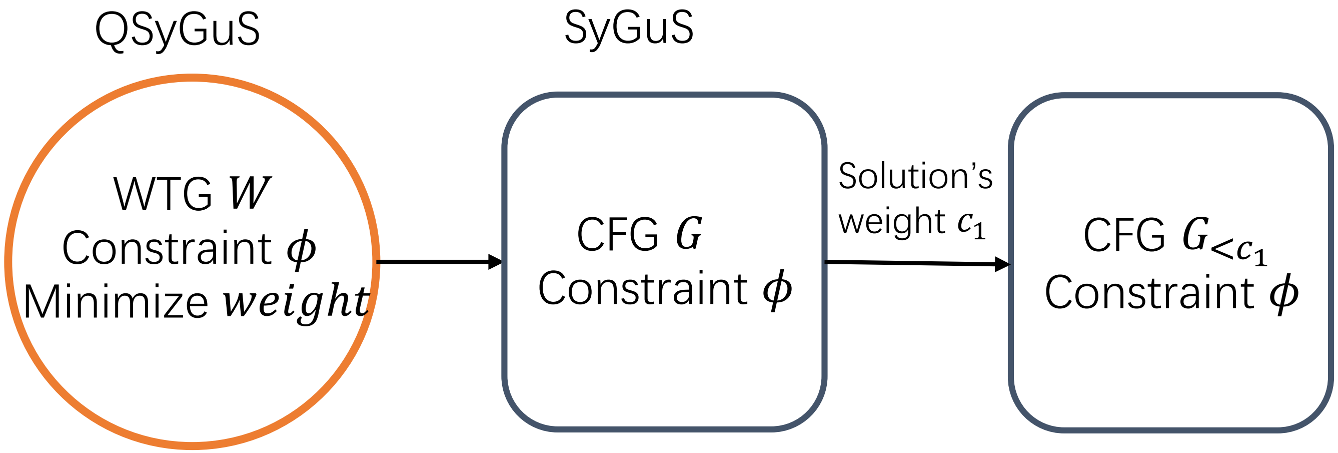
QSyGuS

WTG W
Constraint ϕ
Minimize $weight$

ignore weight

SyGuS

CFG G
Constraint ϕ



QSyGuS

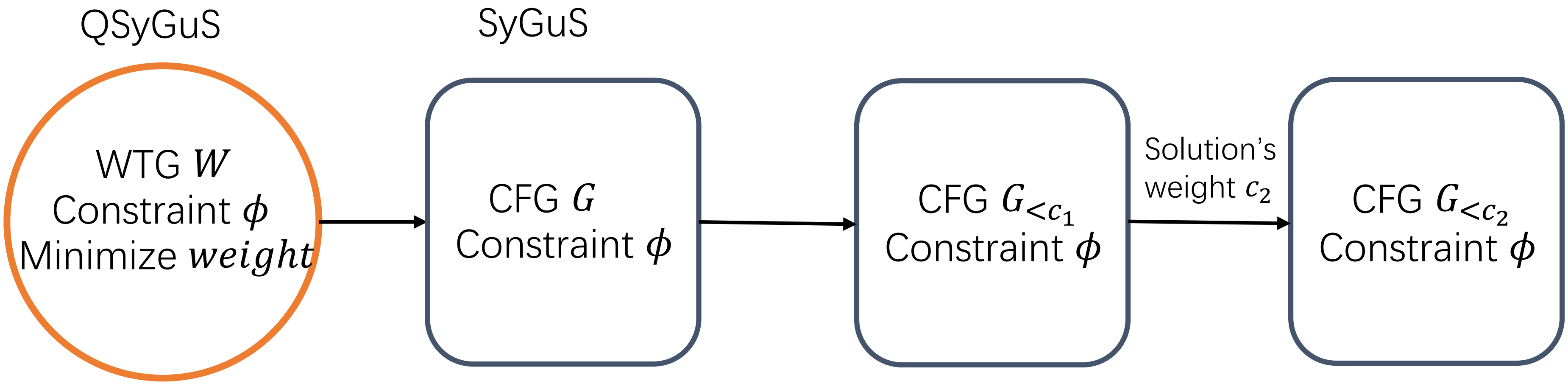
WTG W
Constraint ϕ
Minimize $weight$

SyGuS

CFG G
Constraint ϕ

Solution's
weight c_1

CFG $G_{<c_1}$
Constraint ϕ



QSyGuS

WTG W
Constraint ϕ
Minimize $weight$

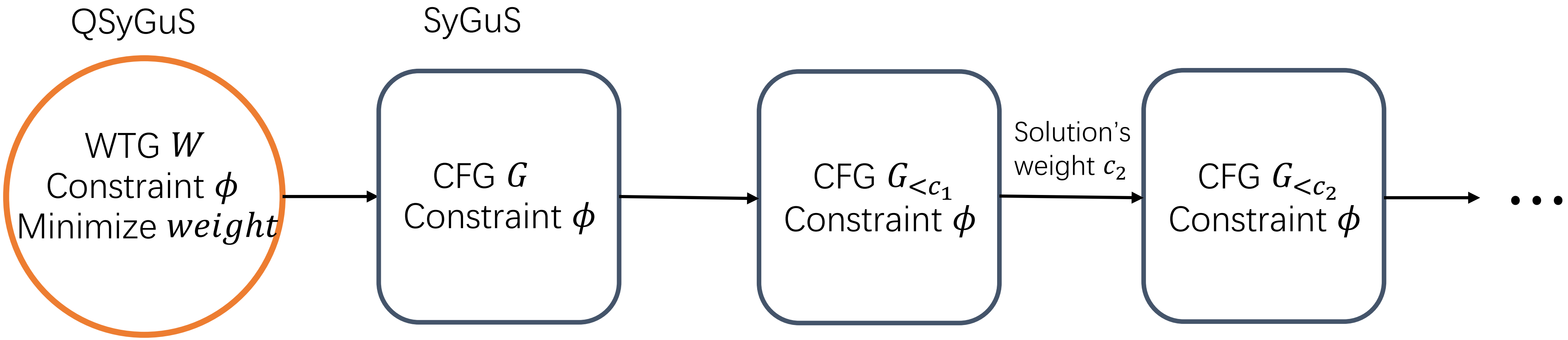
SyGuS

CFG G
Constraint ϕ

CFG $G_{<c_1}$
Constraint ϕ

Solution's
weight c_2

CFG $G_{<c_2}$
Constraint ϕ



QSyGuS

WTG W
Constraint ϕ
Minimize $weight$

SyGuS

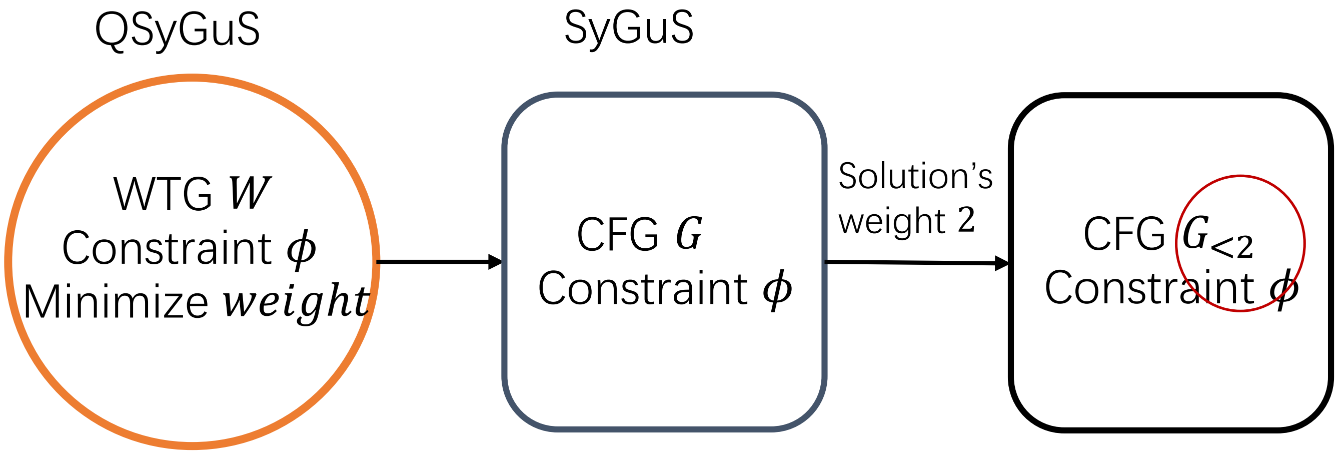
CFG G
Constraint ϕ

CFG $G_{<c_1}$
Constraint ϕ

Solution's weight c_2

CFG $G_{<c_2}$
Constraint ϕ

…



QSyGuS

SyGuS

WTG W
Constraint ϕ
Minimize $weight$

CFG G
Constraint ϕ

Solution's weight 2

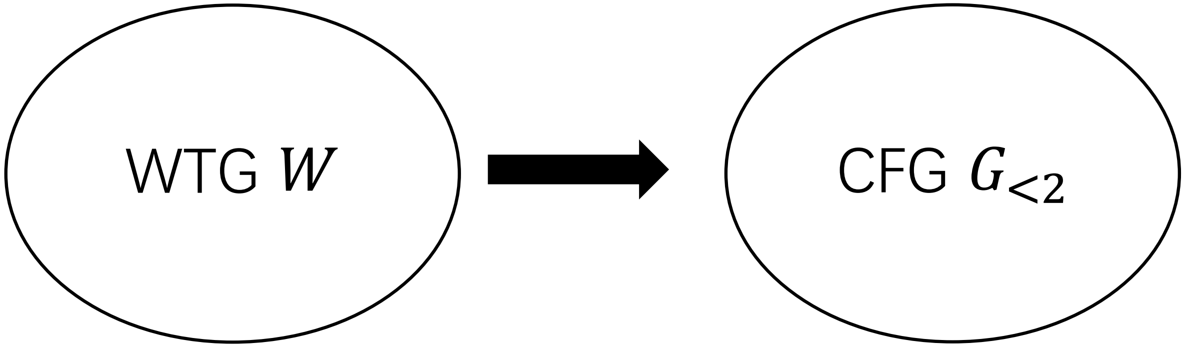
CFG $G_{<2}$
Constraint ϕ

WTG W

CFG $G_{<2}$

Program P has $weight < 2$ iff $G_{<2}$ accept P

Grammar Reduction



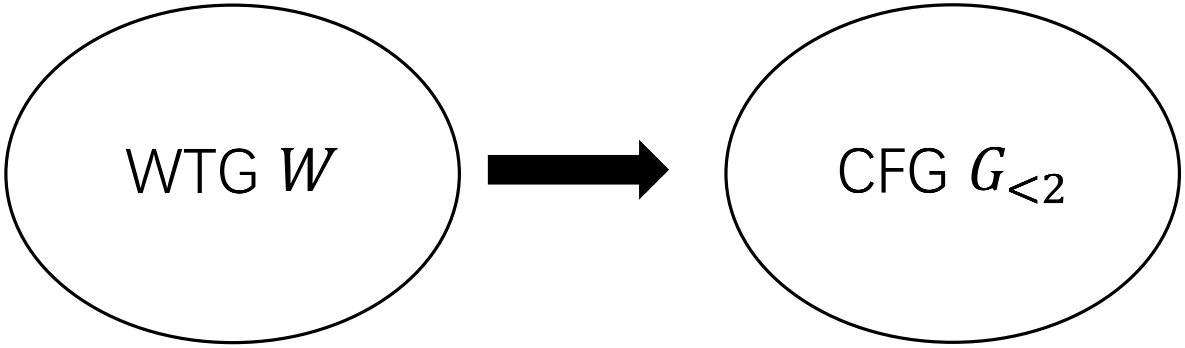
Idea: keep track of the weight in the non-terminals

$$
\begin{aligned}
\text{Start} = {} & \text{Start} + \text{Start} \;/0 \\
& \mid \mathit{if}\,(\text{BExpr})\; \mathit{then}\; \text{Start}\; \mathit{else}\; \text{Start} \;/1 \\
& \mid \text{x}/0 \mid y/0 \mid 0/0 \mid 1/0
\end{aligned}
$$

$$
\begin{aligned}
\text{BExpr} = {} & \text{Start} > \text{Start} \;/0 \\
& \mid \mathit{not}\; \text{BExpr} \;/0 \\
& \mid \text{BExpr}\; \mathit{and}\; \text{BExpr} \;/0
\end{aligned}
$$

$$\Big\downarrow{\scriptstyle\, weight < 2}$$

Grammar Reduction



WTG W

CFG $G_{<2}$

Idea: keep track of the weight in the non-terminals

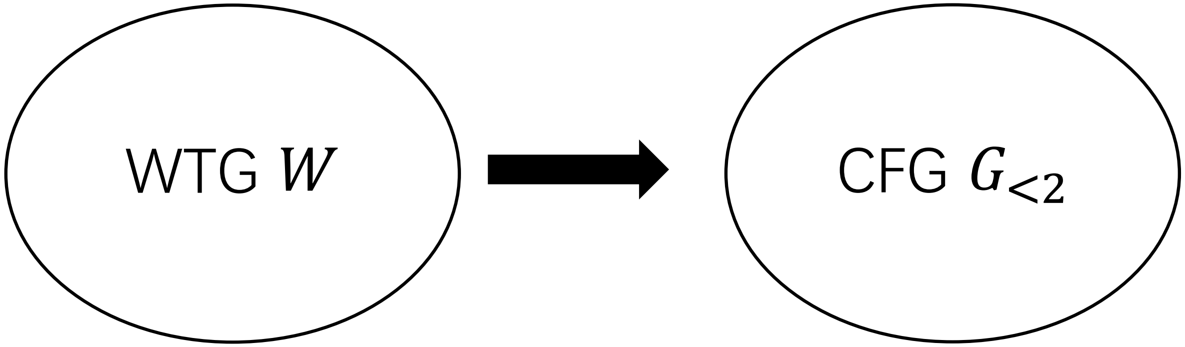
$\text{Start} = \text{Start} + \text{Start}\ /0$
$\quad\mid \mathit{if}\,(\text{BExpr})\ \mathit{then}\ \text{Start}\ \mathit{else}\ \text{Start}\ /1$
$\quad\mid \text{x}/0 \mid y/0 \mid 0/0 \mid 1/0$

$\text{BExpr} = \text{Start} > \text{Start}\ /0$
$\quad\mid \mathit{not}\ \text{BExpr}\ /0$
$\quad\mid \text{BExpr}\ \mathit{and}\ \text{BExpr}\ /0$

$$weight < 2 \ \big\downarrow$$

$$(\text{Start}, < 2) = (\text{Start}, 0) \mid (\text{Start}, 1)$$

Grammar Reduction



WTG W

CFG $G_{<2}$

Idea: keep track of the weight in the non-terminals

$$\begin{aligned}\text{Start} = {} & \text{Start} + \text{Start}\ /0\\ & \mid \mathit{if}\,(\text{BExpr})\ \mathit{then}\ \text{Start}\ \mathit{else}\ \text{Start}\ /1\\ & \mid \text{x}/0 \mid y/0 \mid 0/0 \mid 1/0\end{aligned}$$

$$\begin{aligned}\text{BExpr} = {} & \text{Start} > \text{Start}\ /0\\ & \mid \mathit{not}\ \text{BExpr}\ /0\\ & \mid \text{BExpr}\ \mathit{and}\ \text{BExpr}\ /0\end{aligned}$$

$$weight < 2\ \Big\downarrow$$

$$(\text{Start}, < 2) = (\text{Start}, 0) \mid (\text{Start}, 1)$$

$$\begin{aligned}(\text{Start}, 1) = {} & (\text{Start}, 0) + (\text{Start}, 1) \mid (\text{Start}, 1) + (\text{Start}, 0)\\ & \mid \mathit{if}\,(\text{BExpr}, 0)\ \mathit{then}\ (\text{Start}, 0)\ \mathit{else}\ (\text{Start}, 0)\end{aligned}$$

$$(\text{Start}, 0) = (\text{Start}, 0) + (\text{Start}, 0) \mid x \mid y \mid 0 \mid 1$$

$$\ldots$$

Handling complex weight constraints

Tree grammars are closed under Boolean operations

Minimization → Linear search

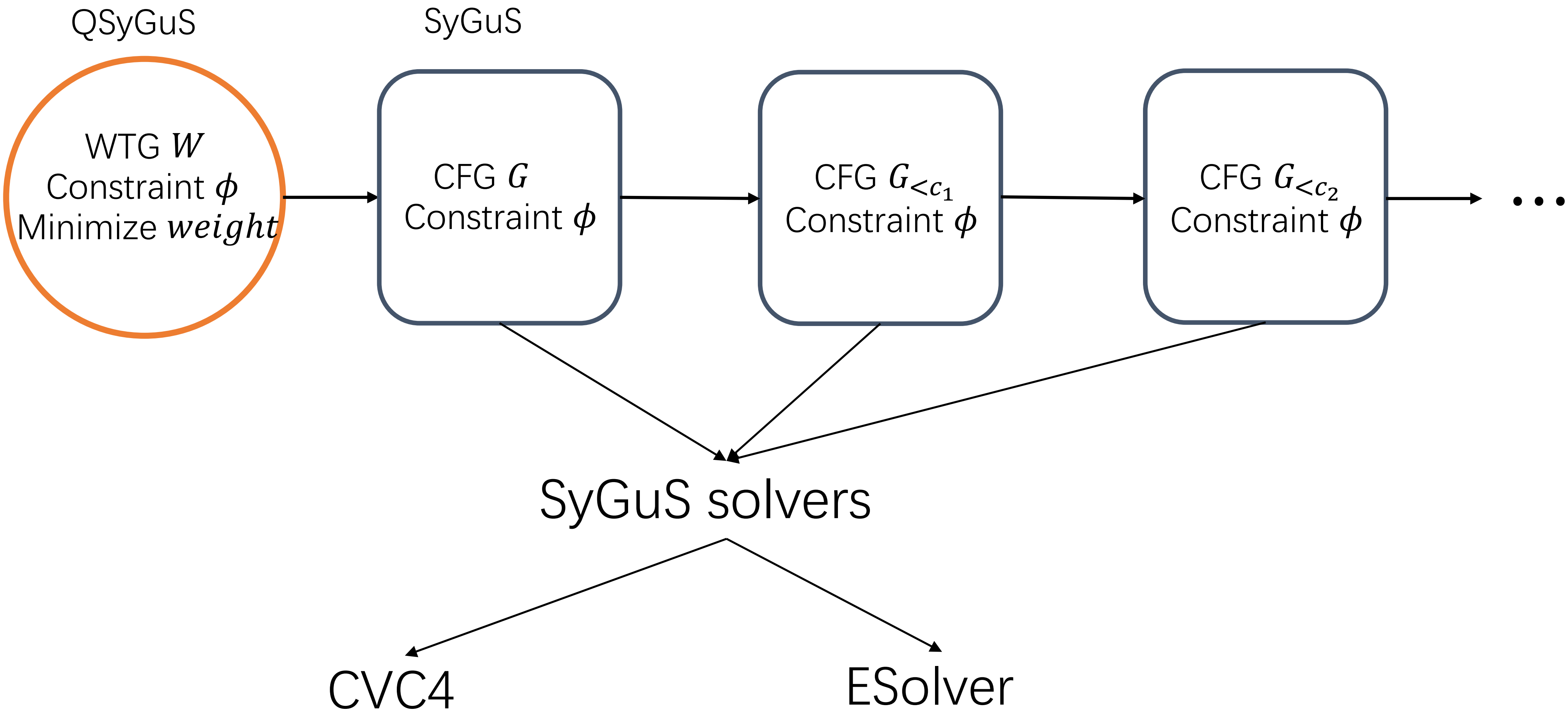
$3 < weight$ → Complement of $G_{<4}$

$2 < weight < 5$ → $G_{<5} \cap G_{>2}$

$3 < weight_1$ and $weight_2 < 0.5$ → $G_{weight_1>3} \cap G_{weight_2<0.5}$

Evaluation

Evaluation



QSyGuS

SyGuS

WTG W
Constraint ϕ
Minimize $weight$

CFG G
Constraint ϕ

CFG $G_{<c_1}$
Constraint ϕ

CFG $G_{<c_2}$
Constraint ϕ

…

SyGuS solvers

CVC4

ESolver

Evaluation

26 Benchmarks taken from SyGuS

1.minimize number of specified operator, minimize solution size

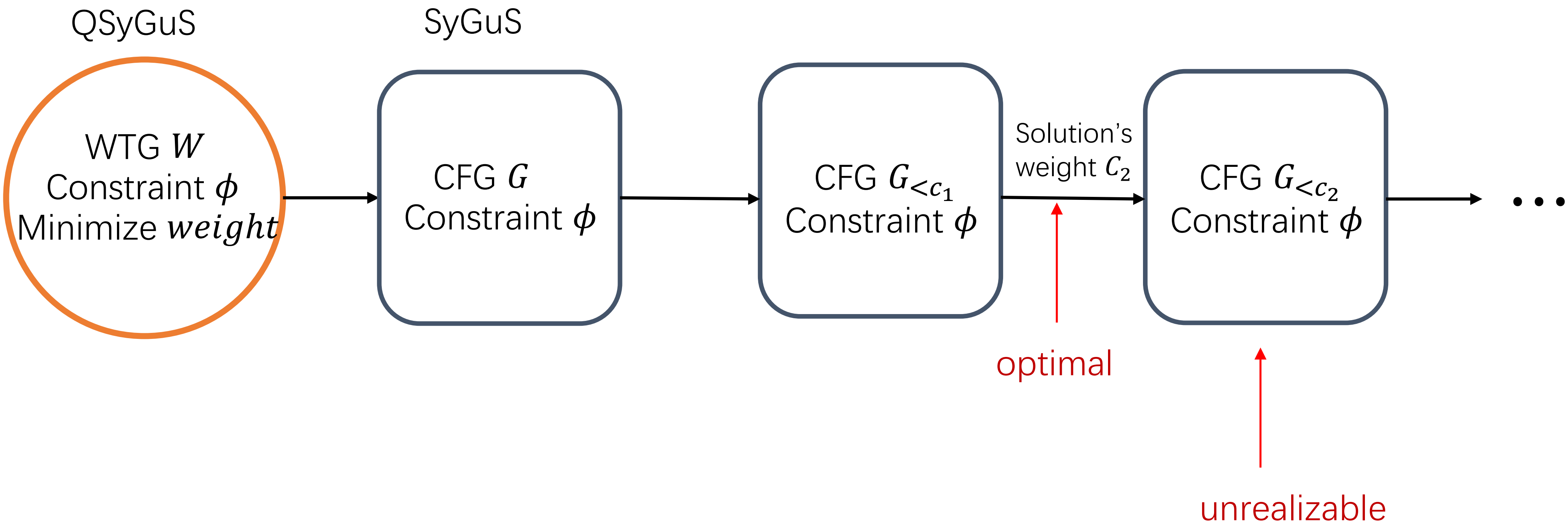
2.maximize solution probability

3.find sorted optimal for (# of specified operators, size)

4.find Pareto optimal for (# of specified operators, size)

Find solution with better cost for 16/26 SyGuS benchmarks
Find optimal in 14/26 (couldn’t prove optimality for 2 benchmarks)
Average time 3.1x Compared to SyGuS

Conclusion



QSyGuS

WTG W
Constraint ϕ
Minimize $weight$

SyGuS

CFG G
Constraint ϕ

CFG $G_{<c_1}$
Constraint ϕ

Solution's weight C_2

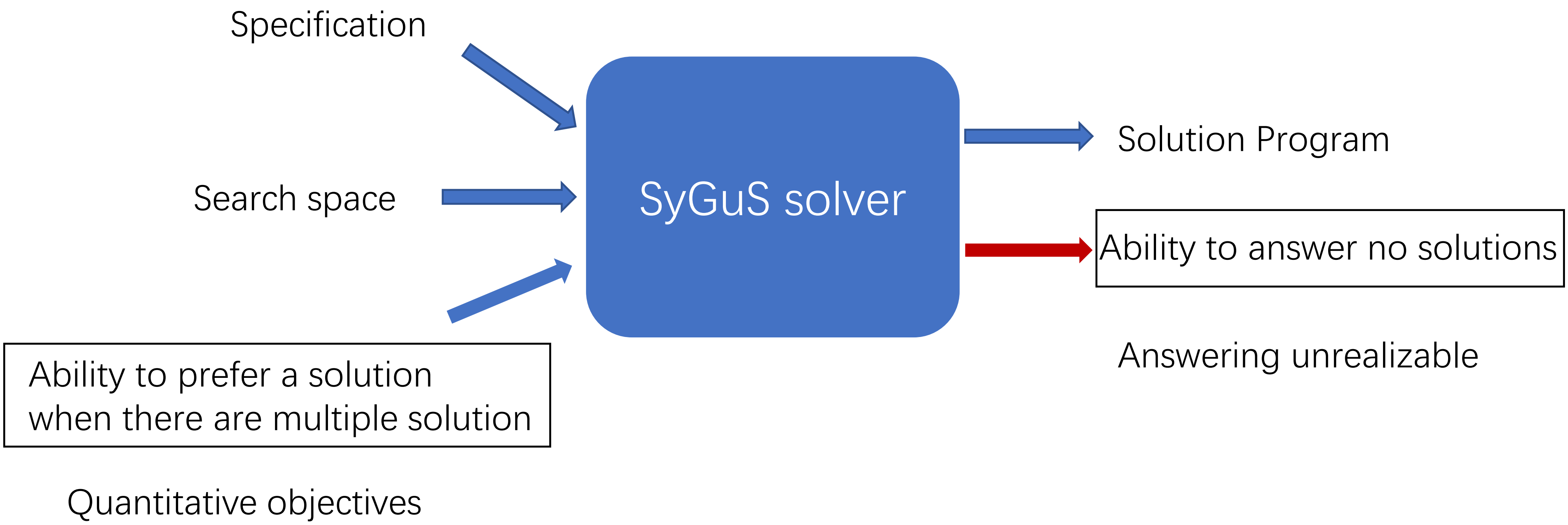
CFG $G_{<c_2}$
Constraint ϕ

…

optimal

unrealizable

Guarantees in SyGuS



Specification

Search space

SyGuS solver

Solution Program

Ability to answer no solutions

Answering unrealizable

Ability to prefer a solution when there are multiple solution

Quantitative objectives

Tue 12:10 come to my CAV talk

Proving Unrealizability for Syntax-Guided Synthesis [CAV19]

A Syntax-Guided Synthesis (SyGuS) is

Specification

$$\varphi(f(x,y),x,y):$$
$$f(x,y) \geq x \wedge$$
$$f(x,y) \geq y \wedge$$
$$(f(x,y) = x \vee f(x,y) = y)$$

Search space G:

$$\begin{aligned} \text{Start} = {} & +(\text{Start}, \text{Start}) \\ & \mid ITE(\text{BExpr}, \text{Start}, \text{Start}) \\ & \mid x \mid y \mid 0 \mid 1 \end{aligned}$$

$$\begin{aligned} \text{BExpr} = {} & Not(\text{BExpr}) \\ & \mid > (\text{Start}, \text{Start}) \\ & \mid And(\text{BExpr}, \text{BExpr}) \end{aligned}$$

Goal: find a program $e \in L(G)$ such that $\forall x, y. \varphi(e, x, y)$

$$\max(x,y) = ITE(>(x,y), x, y)$$

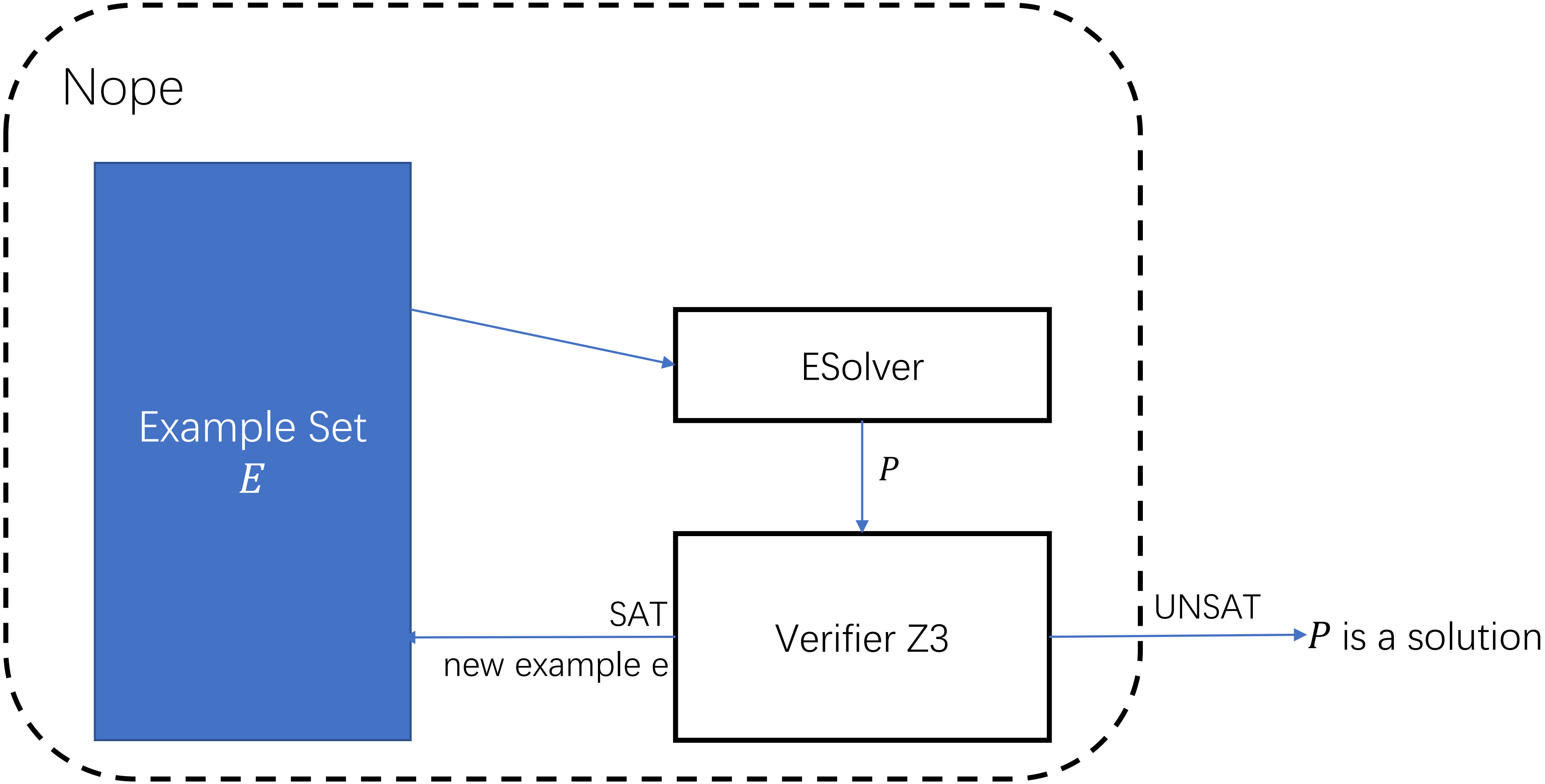
Unrealizable SyGuS Problems

$$\begin{aligned}\text{Start} = {} & +(\text{Start}, \text{Start}) \\ & \mid x \mid y \mid 0 \mid 1\end{aligned}$$

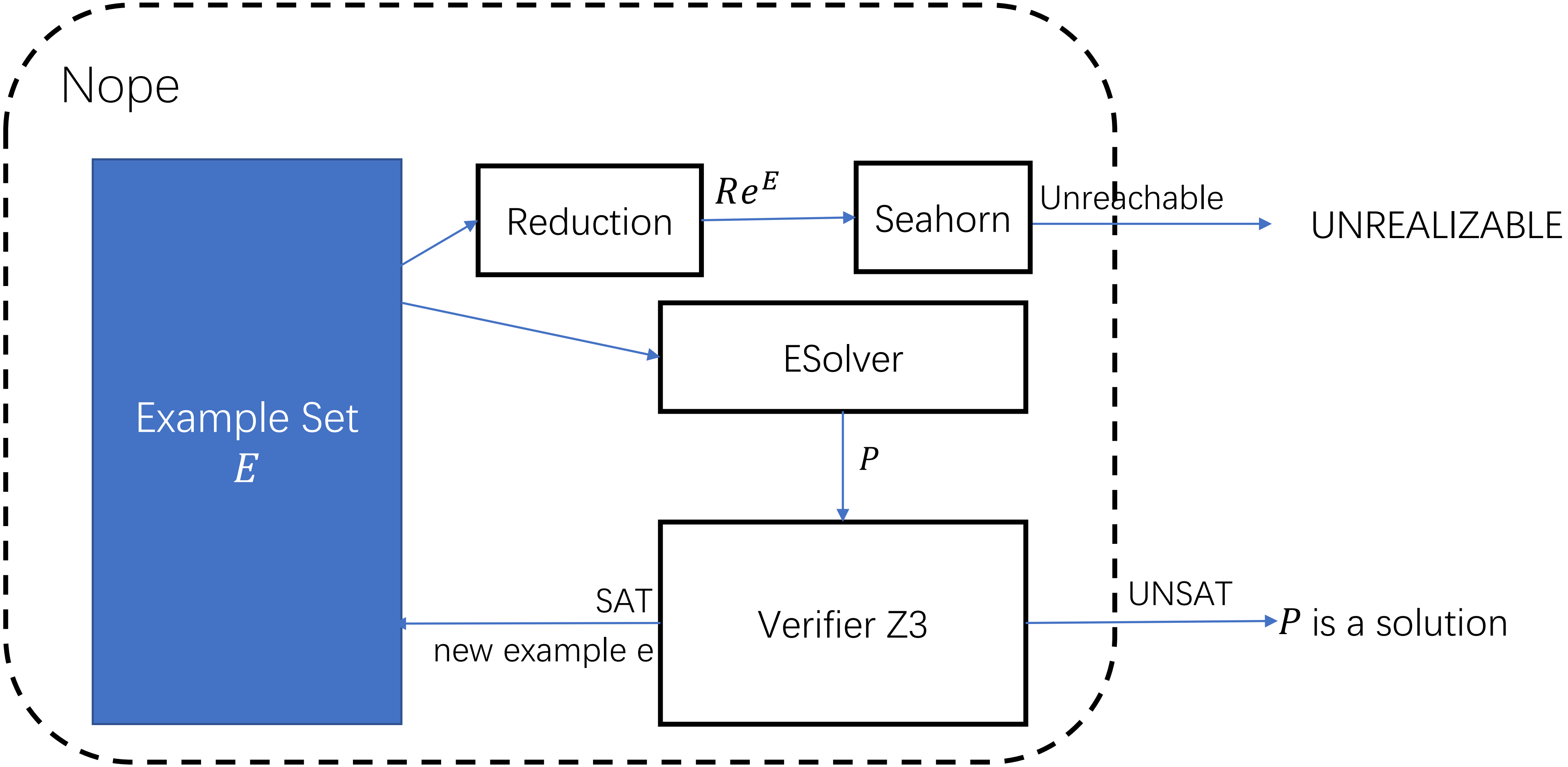
$$\forall x, y.\, \max(x, y) \geq x \wedge \max(x, y) \geq y \wedge (\max(x, y) = x \vee \max(x, y) = y)$$

No Solution

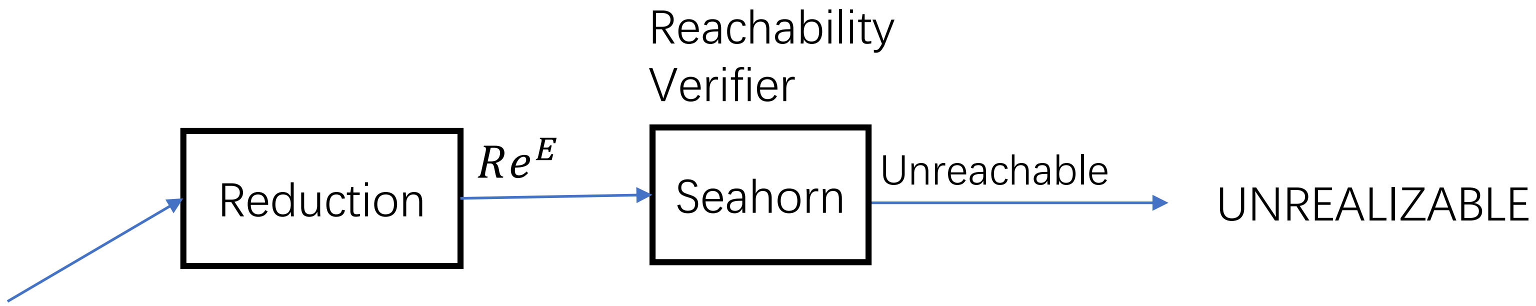
CEGIS-based Frameowrk



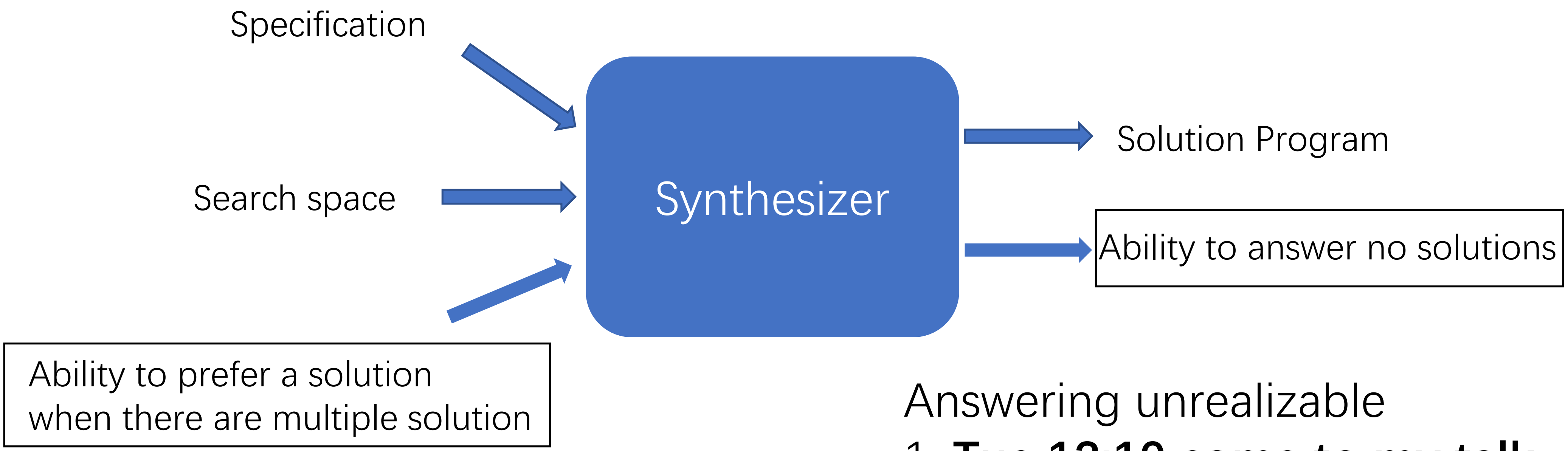
CEGIS-based Frameowrk



SyGuS is unrealizable ↔ reachability problem Re^E is unsatisfiable



Guarantees in Program Synthesis


Specification
Search space
Synthesizer
Solution Program
Ability to answer no solutions
Ability to prefer a solution when there are multiple solution

Answering unrealizable
1. **Tue 12:10 come to my talk**
2. Beyond SyGuS

More quantitative objectives
1.Semantic quantitative objectives
2.Resource bounded synthesis