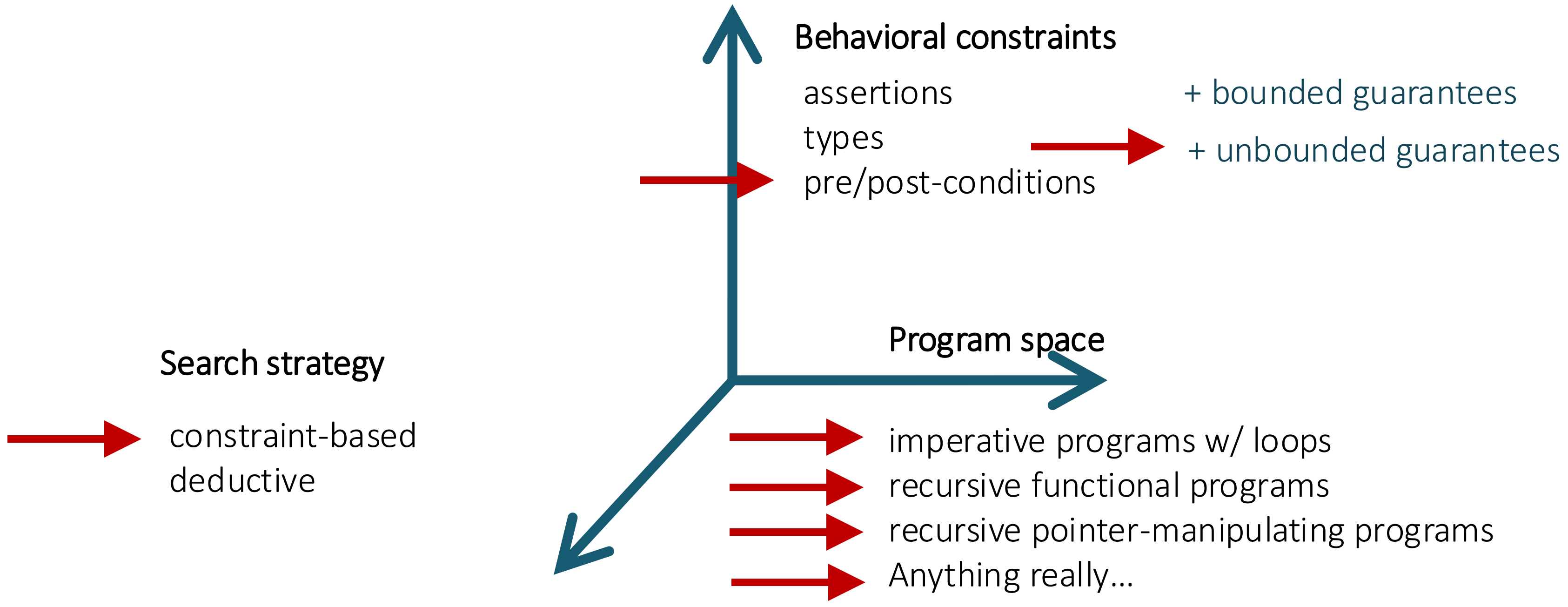
Semantics-Guided Synthesis

Today



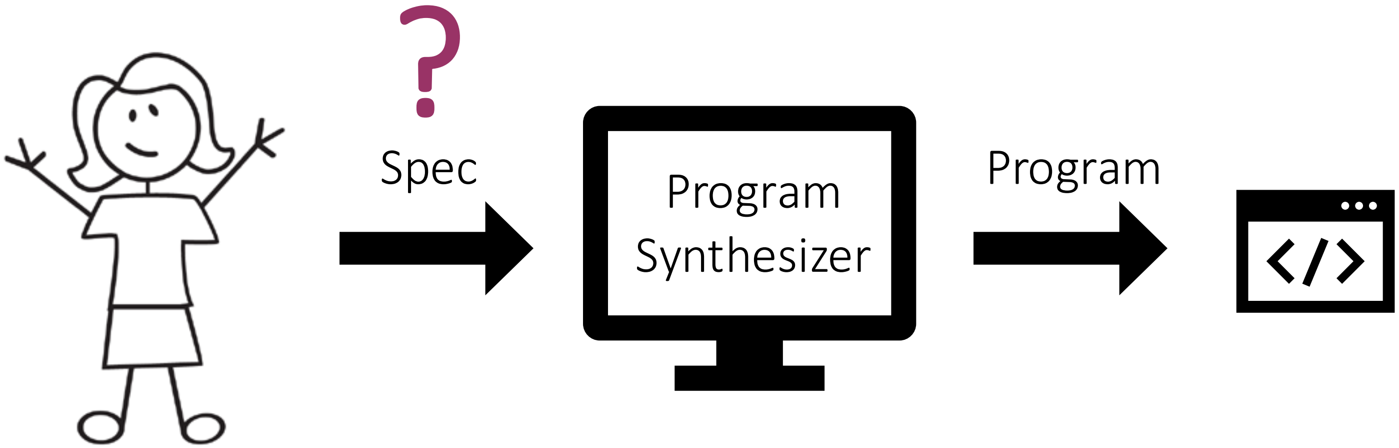
Behavioral constraints
assertions
types
pre/post-conditions
+ bounded guarantees
+ unbounded guarantees
Program space
imperative programs w/ loops
recursive functional programs
recursive pointer-manipulating programs
Anything really…
Search strategy
constraint-based
deductive

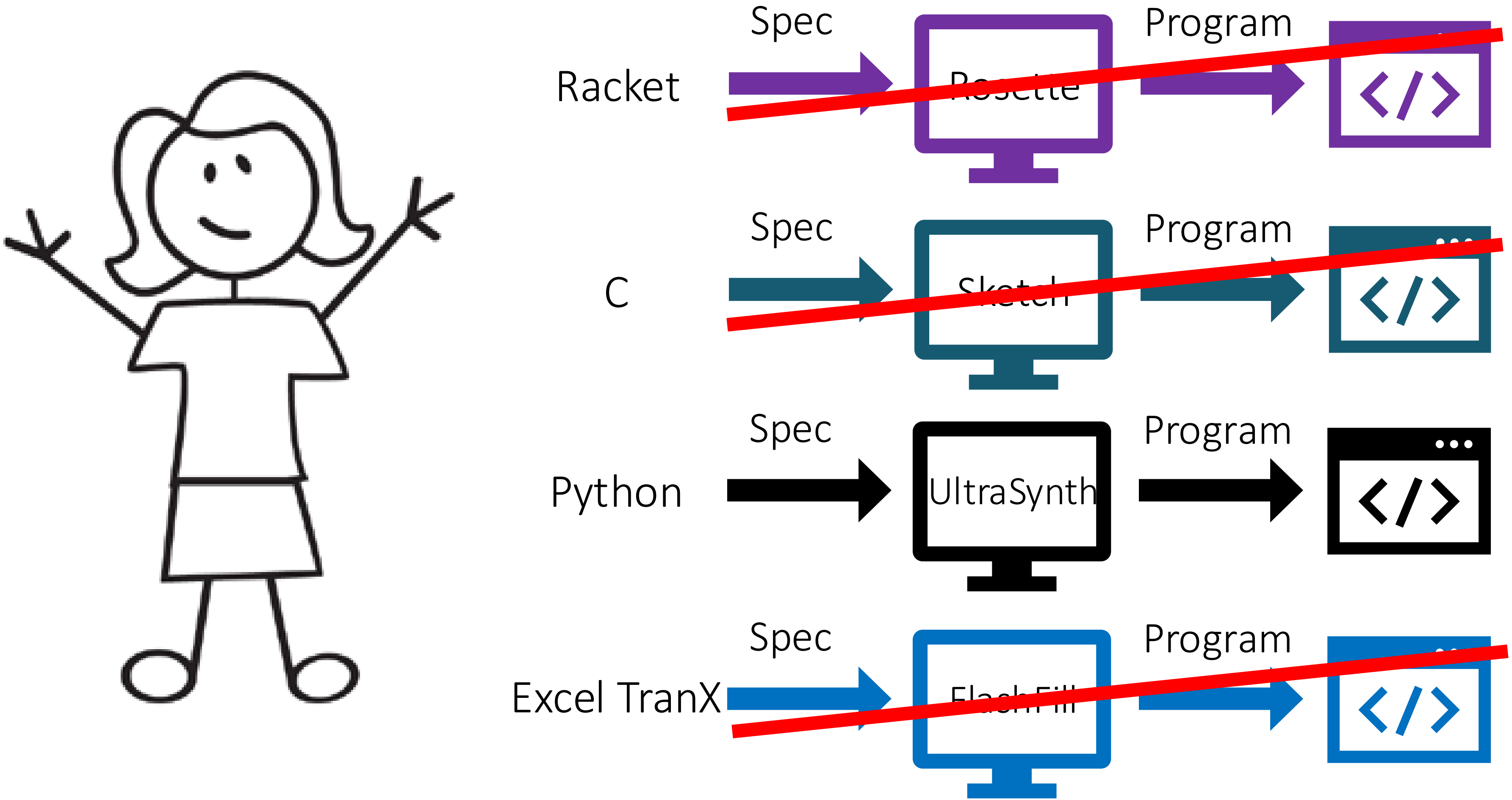
[-1,2,3,10,31,-14,-11]

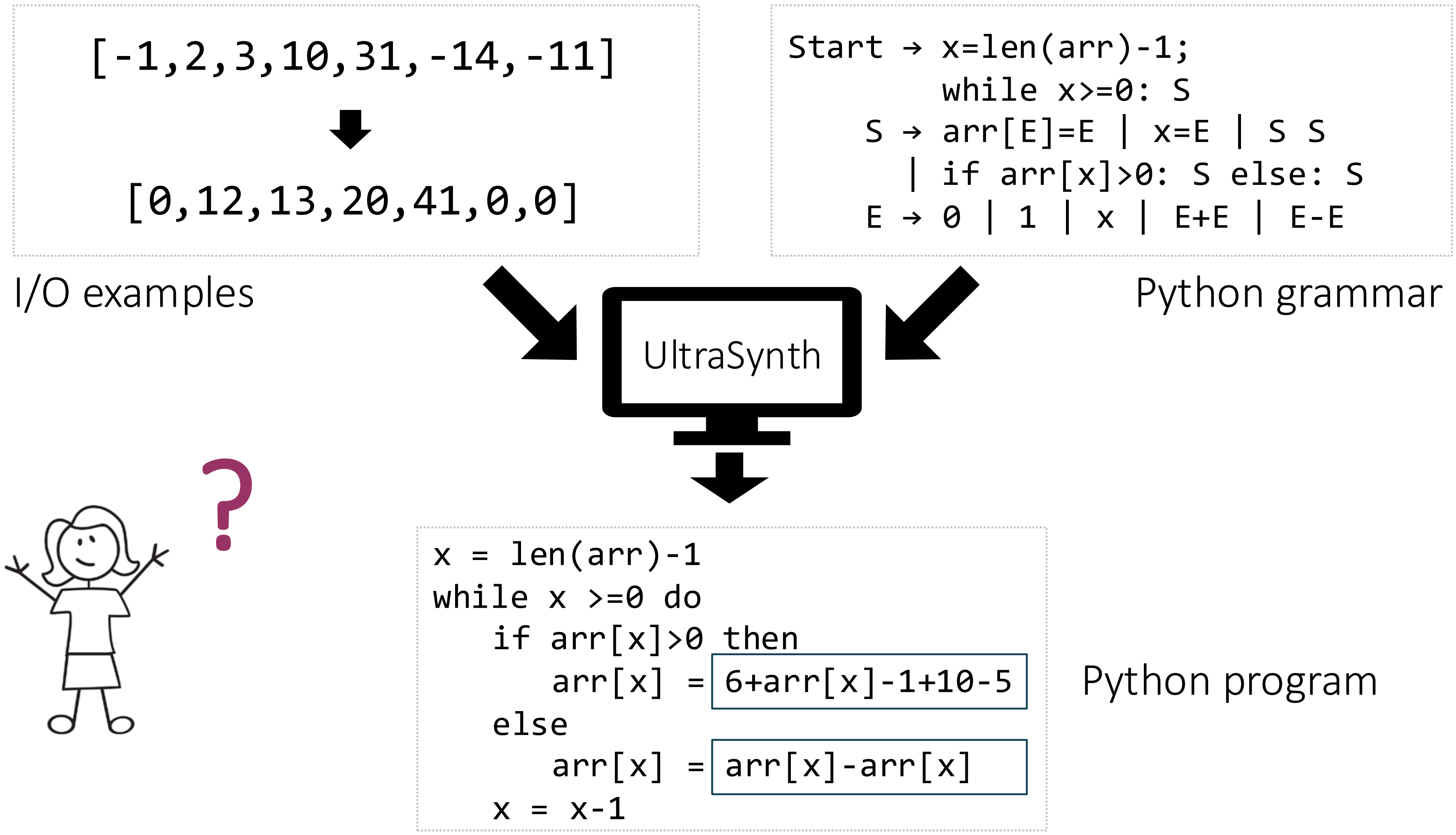
⬇

[0,12,13,20,41,0,0]

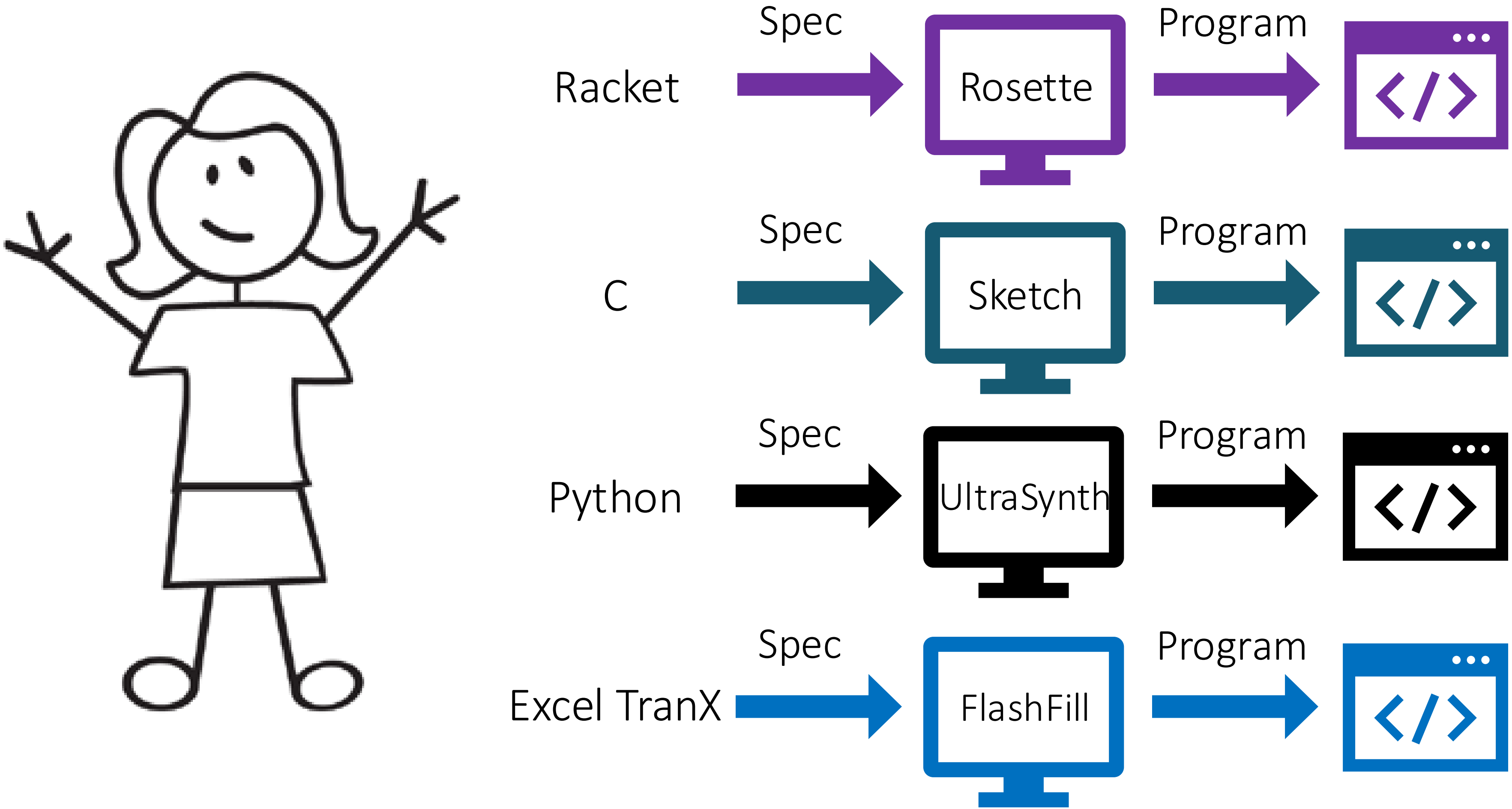
```
x = len(arr)-1 while x >=0:
if arr[x]>0:
    arr[x] = arr[x]+10
else:
    arr[x] = 0
x = x-1
```




Racket
Spec
~~Rosette~~
Program
C
Spec
~~Sketch~~
Program
Python
Spec
UltraSynth
Program
Excel TranX
Spec
~~FlashFill~~
Program


[-1,2,3,10,31,-14,-11]
[0,12,13,20,41,0,0]
I/O examples
Start → x=len(arr)-1;
while x>=0: S
S → arr[E]=E | x=E | S S
| if arr[x]>0: S else: S
E → 0 | 1 | x | E+E | E-E
Python grammar
UltraSynth
?
x = len(arr)-1
while x >=0 do
if arr[x]>0 then
arr[x] = 6+arr[x]-1+10-5
else
arr[x] = arr[x]-arr[x]
x = x-1
Python program

This is not the program I want.
Should I try a different synthesizer?


Racket
Spec
Rosette
Program
C
Spec
Sketch
Program
Python
Spec
UltraSynth
Program
Excel TranX
Spec
FlashFill
Program

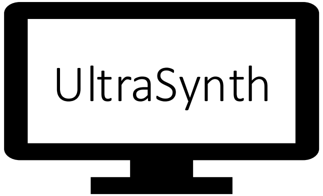
Maybe I’ll hack my way

```
[-1,2,3,10,31,-14,-11]
         ↓
[0,12,13,20,41,0,0]
```

I/O examples

```
Start → x=len(arr)-1;
        while x>=0: S
    S → arr[E]=E | x=E | S S
      | if arr[x]>0: S else: S
    E → 0 | 1 | x | E+E | E-E
```

Python grammar



Fewest minus operators

```
x = len(arr)-1
while x >=0:
    if arr[x]>0:
        arr[x] = 6+arr[x]-1+10-5
    else:
        arr[x] = arr[x]-arr[x]
    x = x-1
```

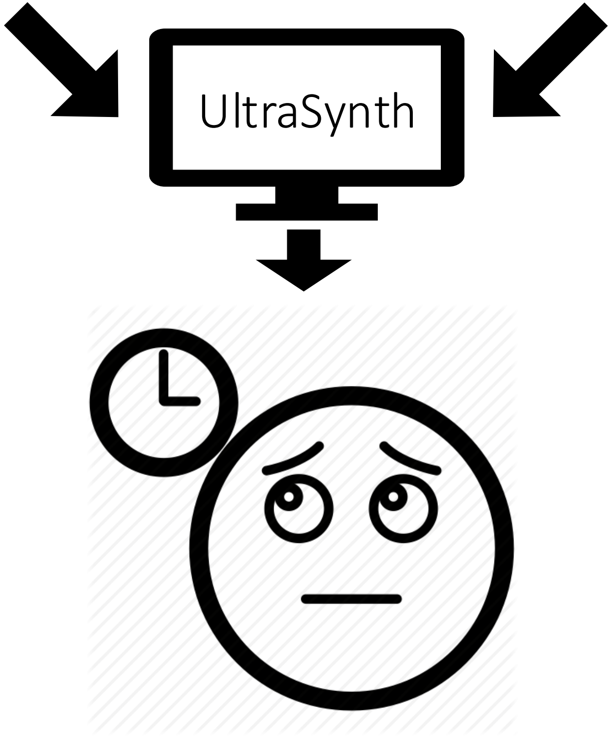
Python program

```
[-1,2,3,10,31,-14,-11]
        ⬇
[0,12,13,20,41,0,0]
```

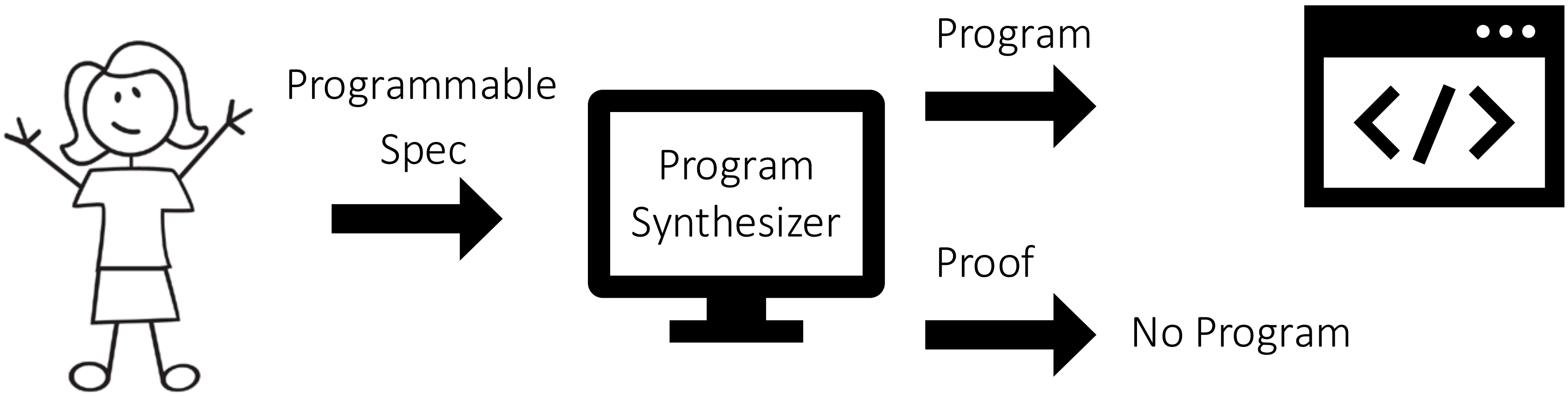
I/O examples

```
Start → x=len(arr)-1;
        while x>=0: S
    S → arr[E]=E | x=E | S S
      | if arr[x]>0: S else: S
    E → 0 | 1 | x | E+E | E-E
```

Python grammar



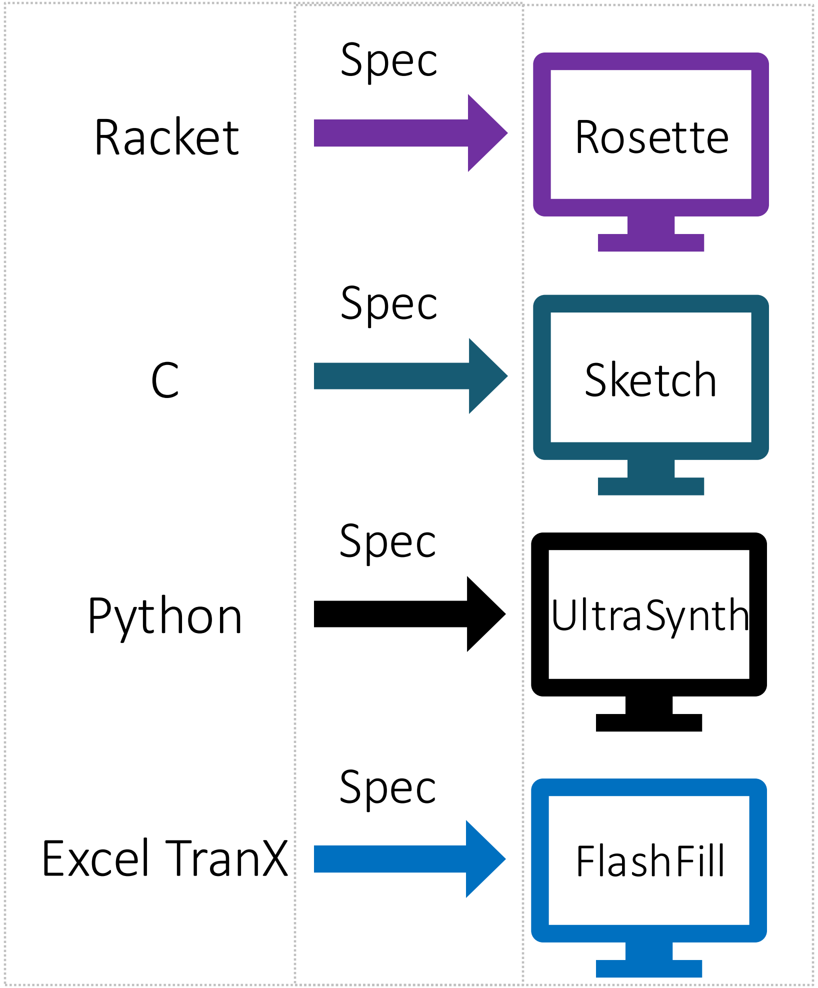

```
x = len(arr)-1;
while x >=0:
    if arr[x]>0:
        arr[x] = arr[x]+10
    else:
        arr[x] = 0
    x = x-1
```


Programmable
Spec
Program
Synthesizer
Program
Proof
No Program

Programmable program synthesis

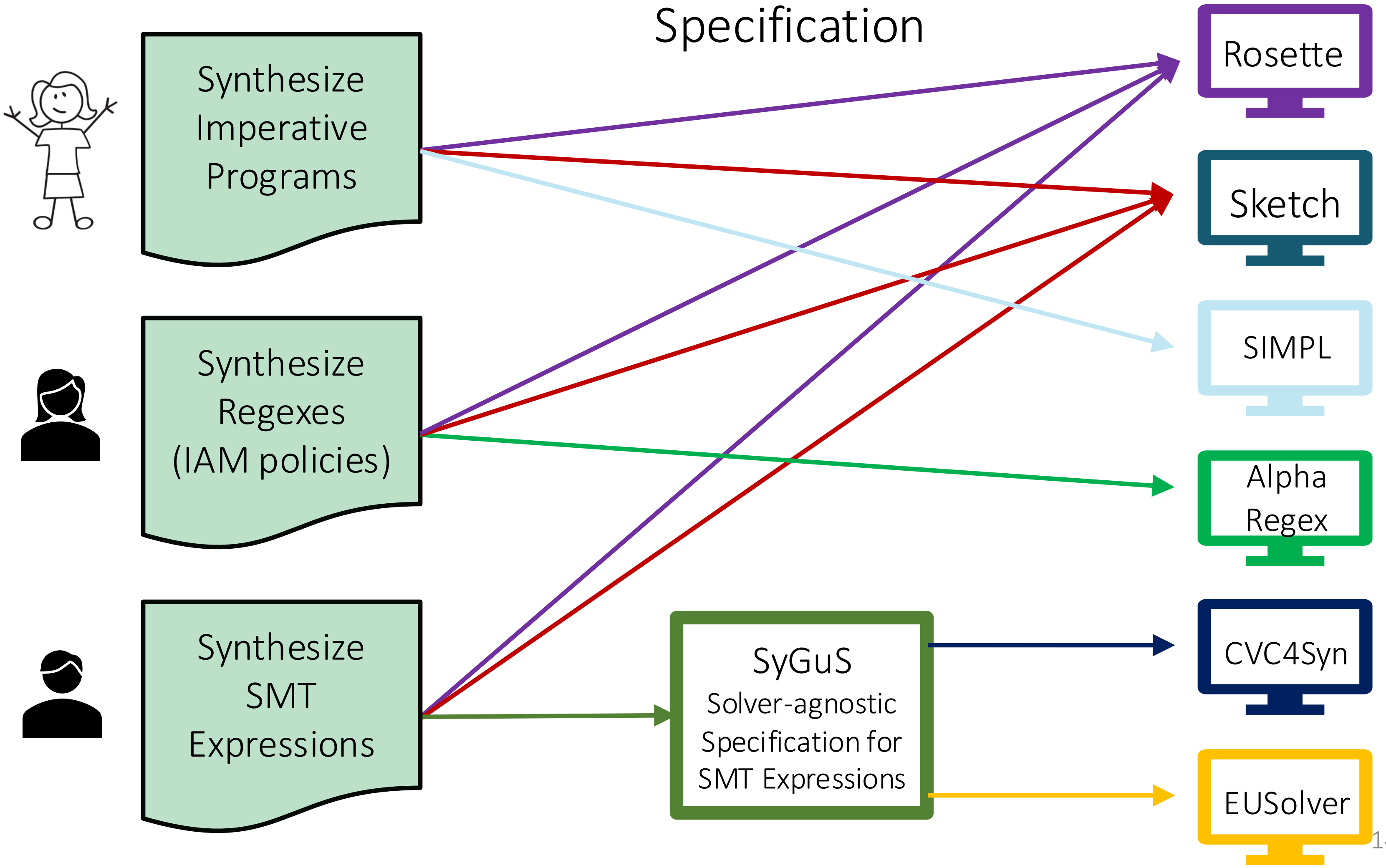
1

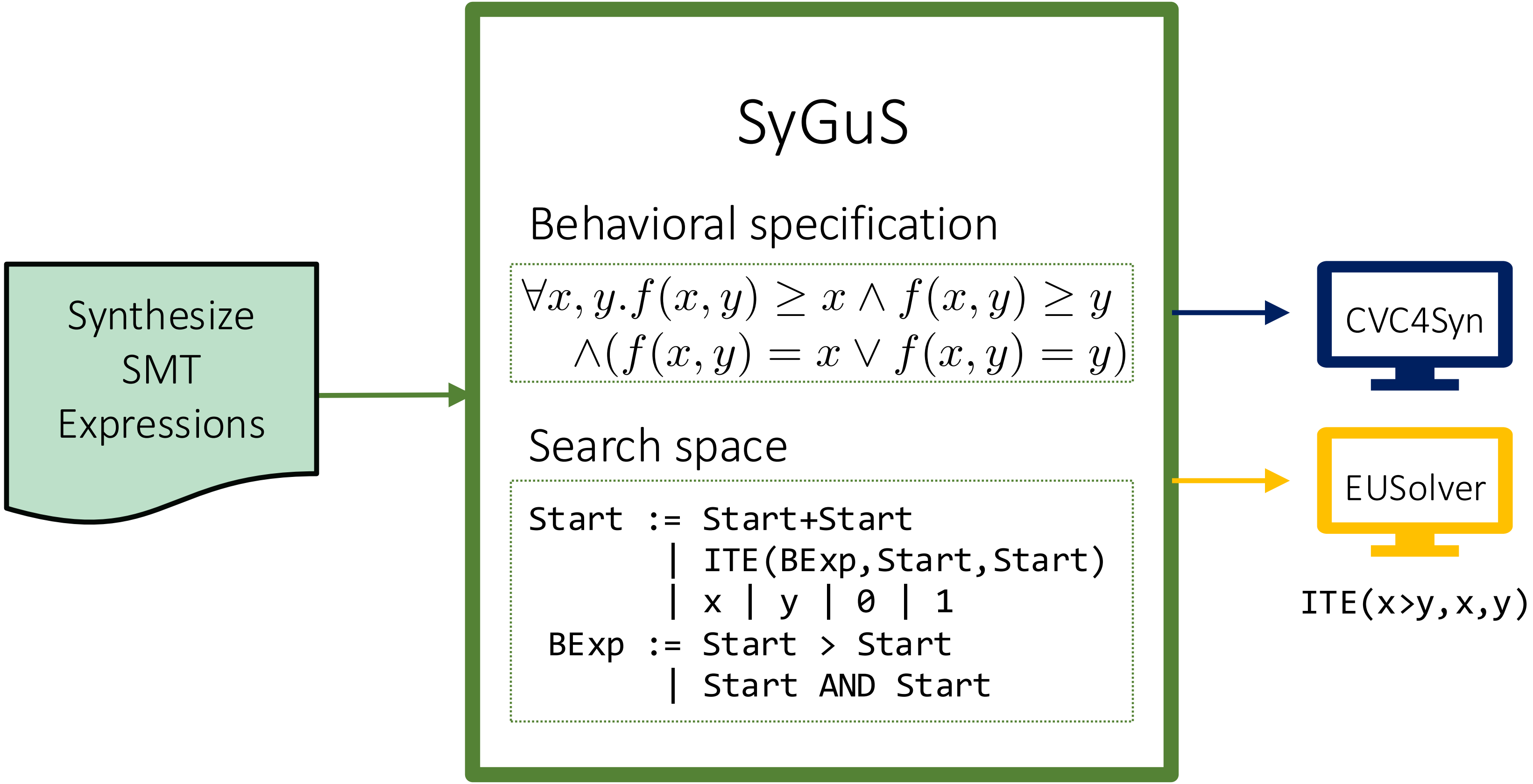
Specification is NOT domain-agnostic



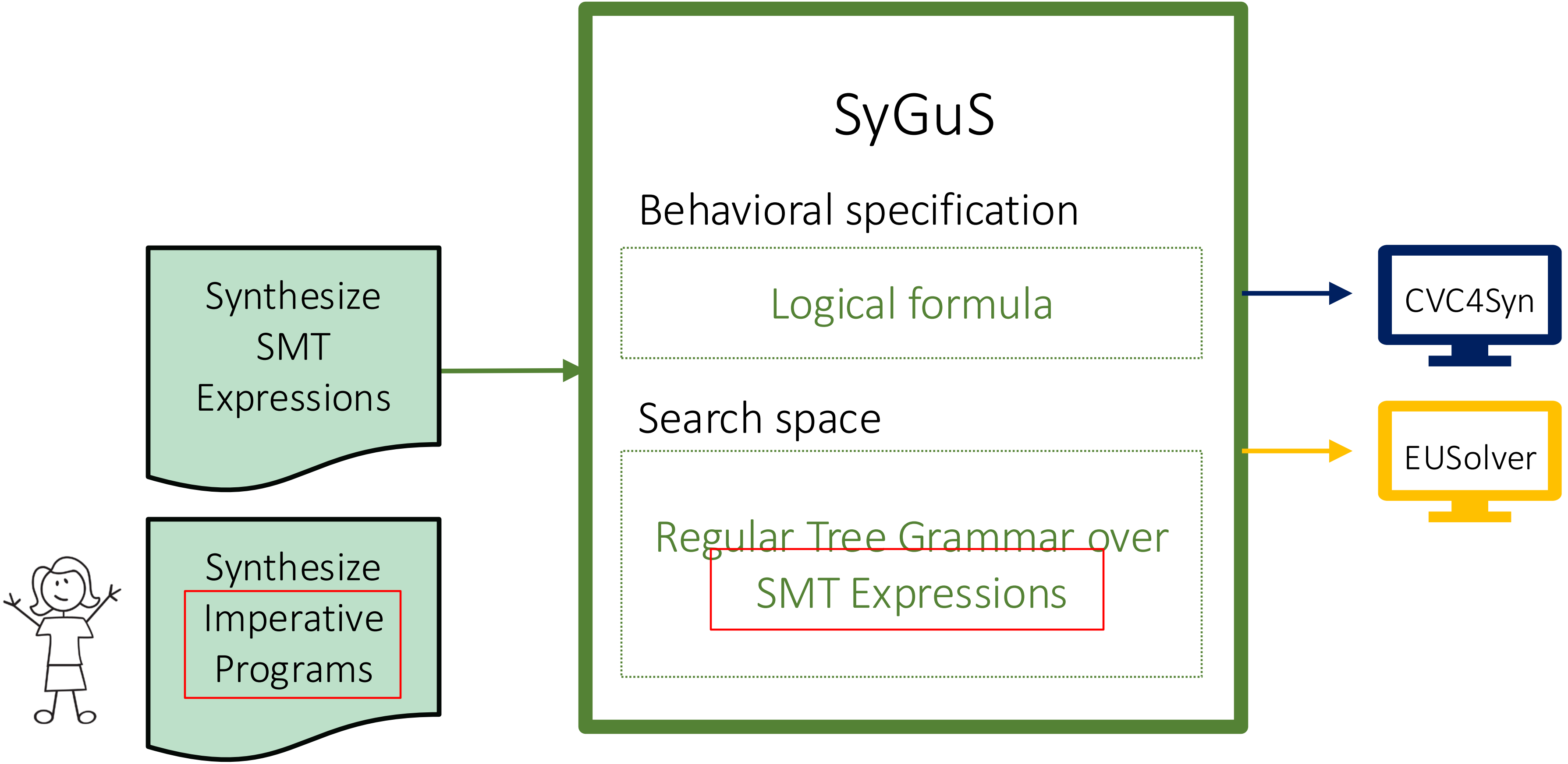
2

Specification is NOT solver-agnostic


Specification
Synthesize Imperative Programs
Synthesize Regexes (IAM policies)
Synthesize SMT Expressions
SyGuS
Solver-agnostic Specification for SMT Expressions
Rosette
Sketch
SIMPL
Alpha Regex
CVC4Syn
EUSolver



Synthesize SMT Expressions
SyGuS
Behavioral specification
$\forall x, y. f(x,y) \geq x \wedge f(x,y) \geq y \wedge (f(x,y) = x \vee f(x,y) = y)$
Search space
Start := Start+Start
| ITE(BExp,Start,Start)
| x | y | 0 | 1
BExp := Start > Start
| Start AND Start
CVC4Syn
EUSolver
ITE(x>y,x,y)


Synthesize
SMT
Expressions
Synthesize
Imperative
Programs
SyGuS
Behavioral specification
Logical formula
Search space
Regular Tree Grammar over
SMT Expressions
CVC4Syn
EUSolver

1

Specification is
domain-agnostic

No

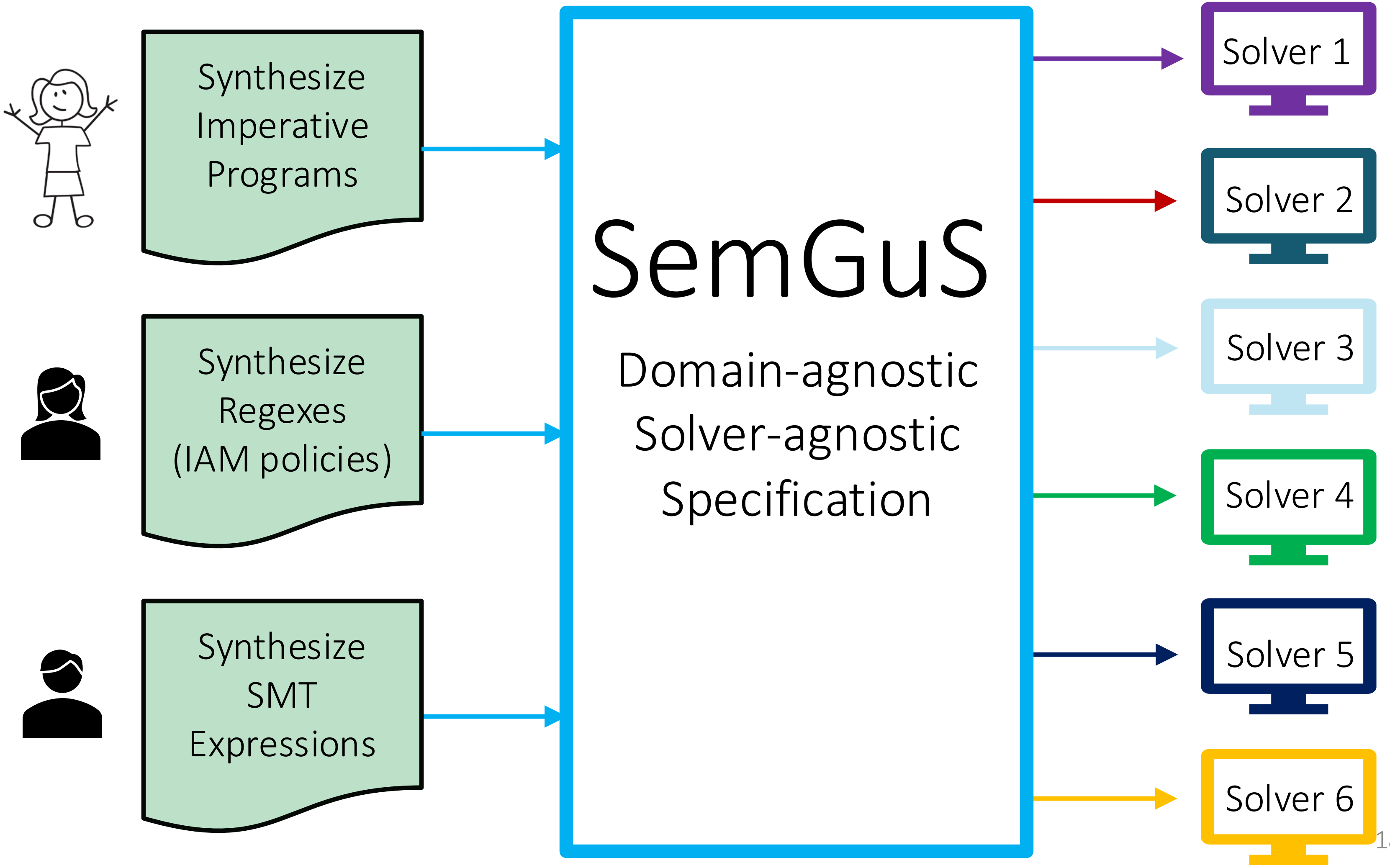
Only supports SMT expressions
with standardized semantics

2

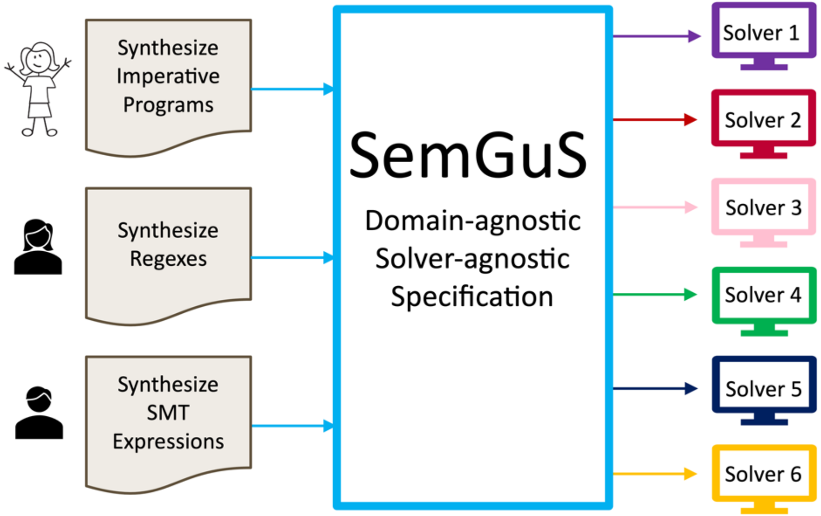
Specification is
solver-agnostic

Yes

Logic and Formal Languages


Synthesize Imperative Programs
Synthesize Regexes (IAM policies)
Synthesize SMT Expressions
SemGuS
Domain-agnostic
Solver-agnostic
Specification
Solver 1
Solver 2
Solver 3
Solver 4
Solver 5
Solver 6

Why is this good?



Programmer

- Uses same spec regardless of domain
- Does not choose a solver a priori

Solvers

- Operate over unified format
- Can be reused and interoperate

Benchmarks

- Can be standardized
- Enable solver competitions

Today

Programmable program synthesis

Semantics-guided synthesis (SemGuS)

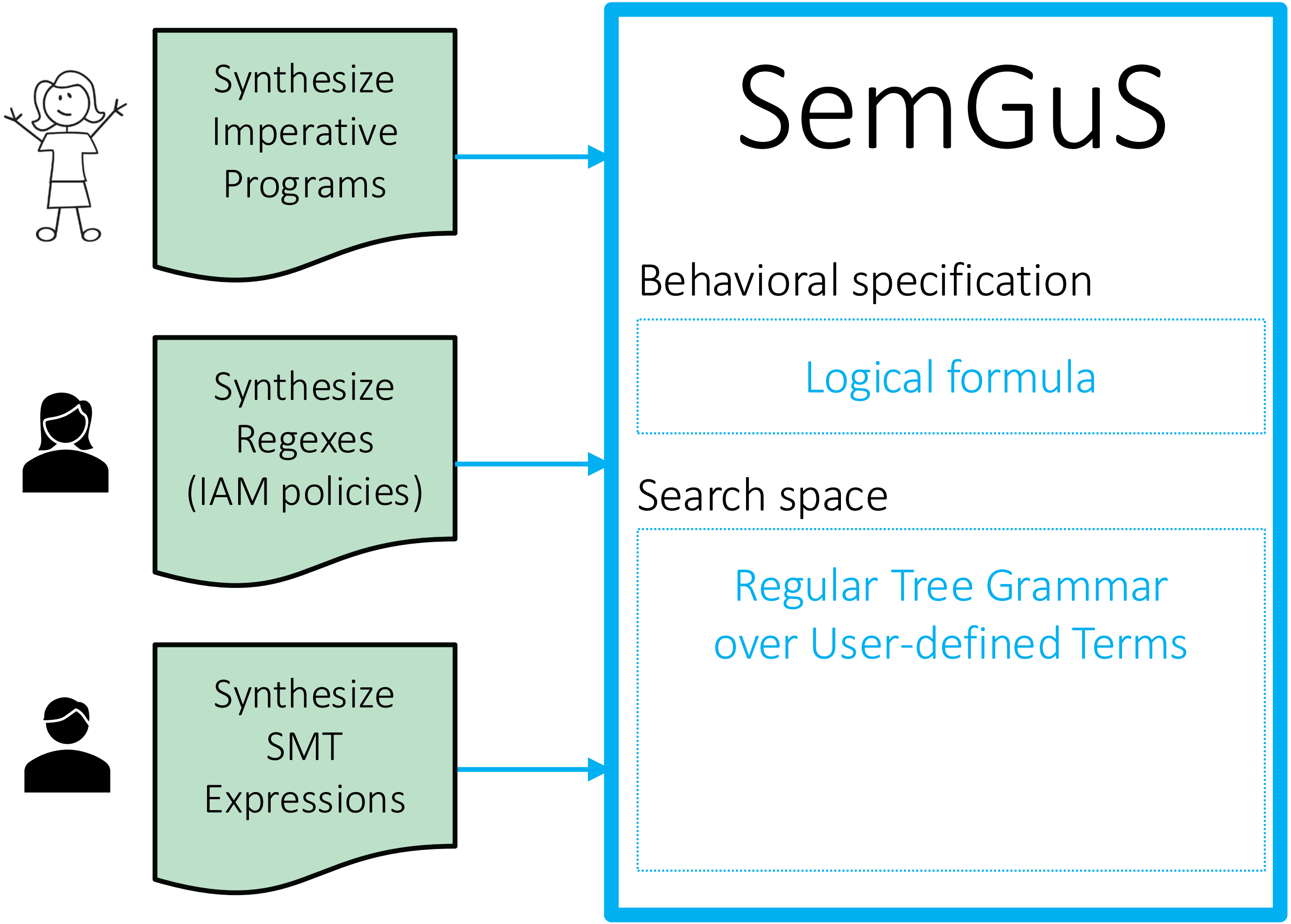
Solving SemGuS problems

The power of programmable frameworks

Some experimental results

SEMANTICS-GUIDED SYNTHESIS

How does it work


Synthesize
Imperative
Programs
Synthesize
Regexes
(IAM policies)
Synthesize
SMT
Expressions
SemGuS
Behavioral specification
Logical formula
Search space
Regular Tree Grammar
over User-defined Terms

SemGuS Problem

Language Syntax	Language Semantics	Behavioral Spec
Regular Tree Grammar	Constrained Horn Clauses	Constraints / Examples

$$Start ::= While\ B\ do\ S$$
$$B ::= E < E$$
$$E ::= x \mid y \mid E \,\&\, E \mid (E \mid E)$$
$$S ::= S; S \mid x \coloneqq E \mid y \coloneqq E$$

SemGuS Problem

Language Syntax	Language Semantics	Behavioral Spec
Regular Tree Grammar	Constrained Horn Clauses	Constraints / Examples

$Start ::= While\ B\ do\ S$
$B ::= E < E$
$E ::= x \mid y \mid E\ \&\ E \mid (E \mid E)$
$S ::= S; S \mid x := E \mid y := E$

?

$Start ::= While\ B\ do\ S$

$B ::= E < E$

$E ::= x \mid y \mid E \,\&\, E \mid (E \mid E)$

$S ::= S; S \mid x \coloneqq E \mid y \coloneqq E$

Every $Term$ in S has type $State \rightarrow State$

$$[\![\, x \coloneqq x\&y \,]\!]\begin{pmatrix} x & y \\ 1, & 0 \end{pmatrix} = \begin{pmatrix} x & y \\ 0, & 0 \end{pmatrix}$$

$$Sem_S \subseteq State \times Term \times State$$

$$Sem_S\left(\begin{pmatrix} x & y \\ 1, & 0 \end{pmatrix}, x \coloneqq x\&y, \begin{pmatrix} x & y \\ 0, & 0 \end{pmatrix}\right)$$

$$Sem_S(\Gamma, t_S, \Gamma') \Leftrightarrow [\![\, t_S \,]\!](\Gamma) = \Gamma'$$

$S ::= x := E$

$$\frac{[\![t_e]\!](\Gamma) = v \quad \Gamma_1 = \Gamma[x \mapsto v]}{[\![x := t_e]\!](\Gamma) = \Gamma_1}$$

$$\frac{\boxed{Sem_E(\Gamma, t_e, v) \quad \Gamma_1 = \Gamma[x \mapsto v]}}{\boxed{Sem_S(\Gamma, x := t_e, \Gamma_1)}}$$

Constrained Horn Clause

$$\forall \Gamma, \Gamma_1, t_e, v. \boxed{\overbrace{Sem_E(\Gamma, t_e, v)}^{\text{Relation Part}} \wedge \overbrace{\Gamma_1 = \Gamma[x \mapsto v]}^{\text{First-Order Constraint}} \Rightarrow} \boxed{\overbrace{Sem_S(\Gamma, x := t_e, \Gamma_1)}^{\text{Formula Head}}}$$

$Start ::= While\ B\ do\ S$

$$\frac{Sem_{\boxed{B}}(\Gamma, t_b, v) \quad v = \boxed{True} \quad Sem_{\boxed{S}}(\Gamma, t_s, \Gamma_1) \quad Sem_{\boxed{Start}}(\Gamma_1, While\ t_b\ do\ t_s, \Gamma_2)}{Sem_{\boxed{Start}}(\Gamma, While\ t_b\ do\ t_s, \Gamma_2)}$$

Nonterminals can have different signatures

Every *Term* in B has type $State \rightarrow Bool$

$$\frac{Sem_B(\Gamma, t_b, v) \quad v = \boxed{False}}{Sem_{Start}(\Gamma, While\ t_b\ do\ t_s, \Gamma)}$$

SemGuS Problem

Language Syntax

Regular Tree Grammar

Language Semantics

Constrained Horn Clauses

Behavioral Spec

Constraints / Examples

$Start ::= While\ B\ do\ S$
$B ::= E < E$
$E ::= x \mid y \mid E\ \&\ E \mid (E \mid E)$
$S ::= S; S \mid x \coloneqq E \mid y \coloneqq E$

$$\frac{Sem_E(\Gamma, t_e, v)\quad \Gamma_1 = \Gamma[x \mapsto v]}{Sem_S(\Gamma, x \coloneqq t_e, \Gamma_1)}$$

Examples:
$f(6, 9) = (15, \_)$

Solution: a program in the **grammar** that when evaluated according to the **semantics** satisfies the behavioral **specification**

1

Specification is

domain-agnostic

Yes

Programmable semantics

2

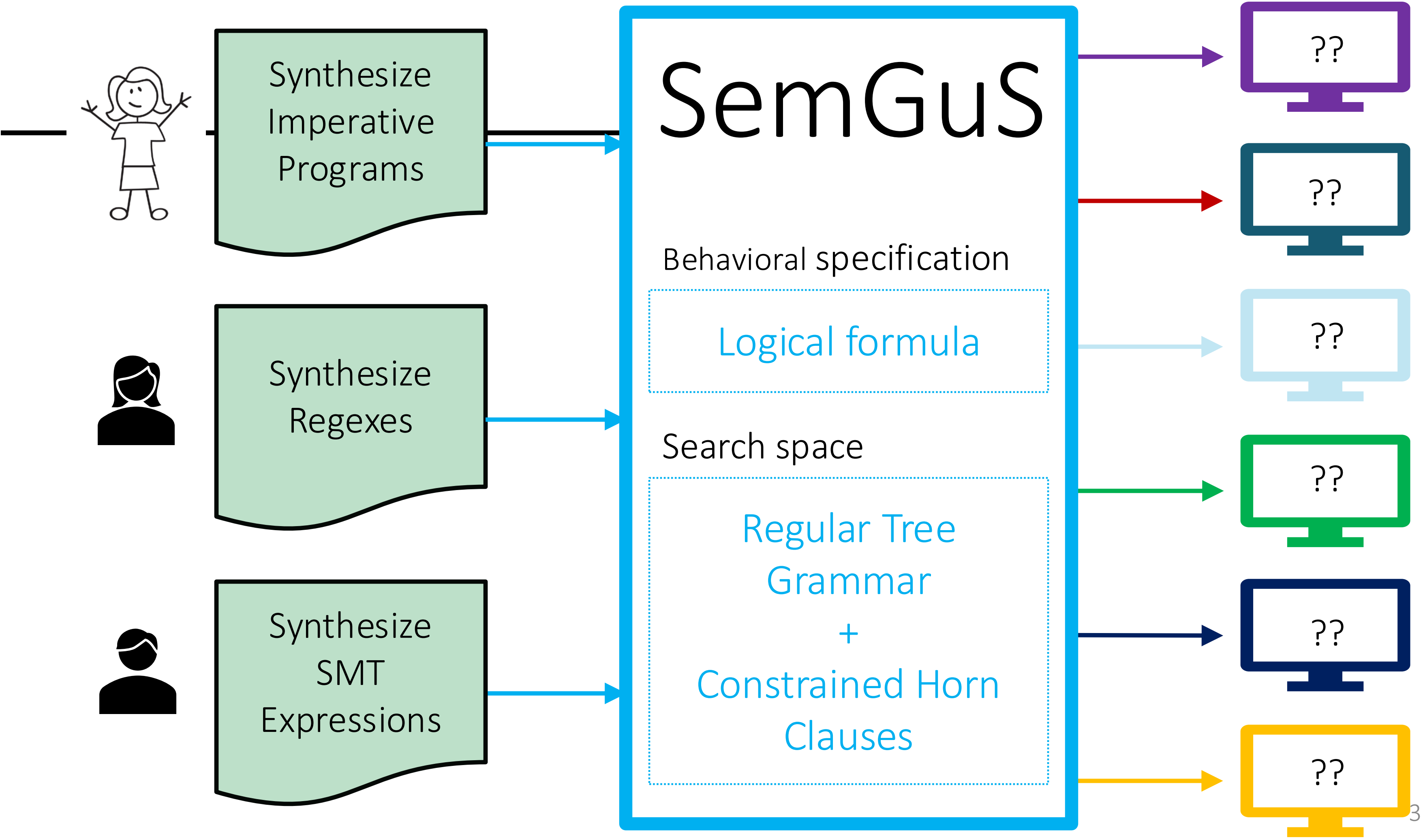
Specification is

solver-agnostic

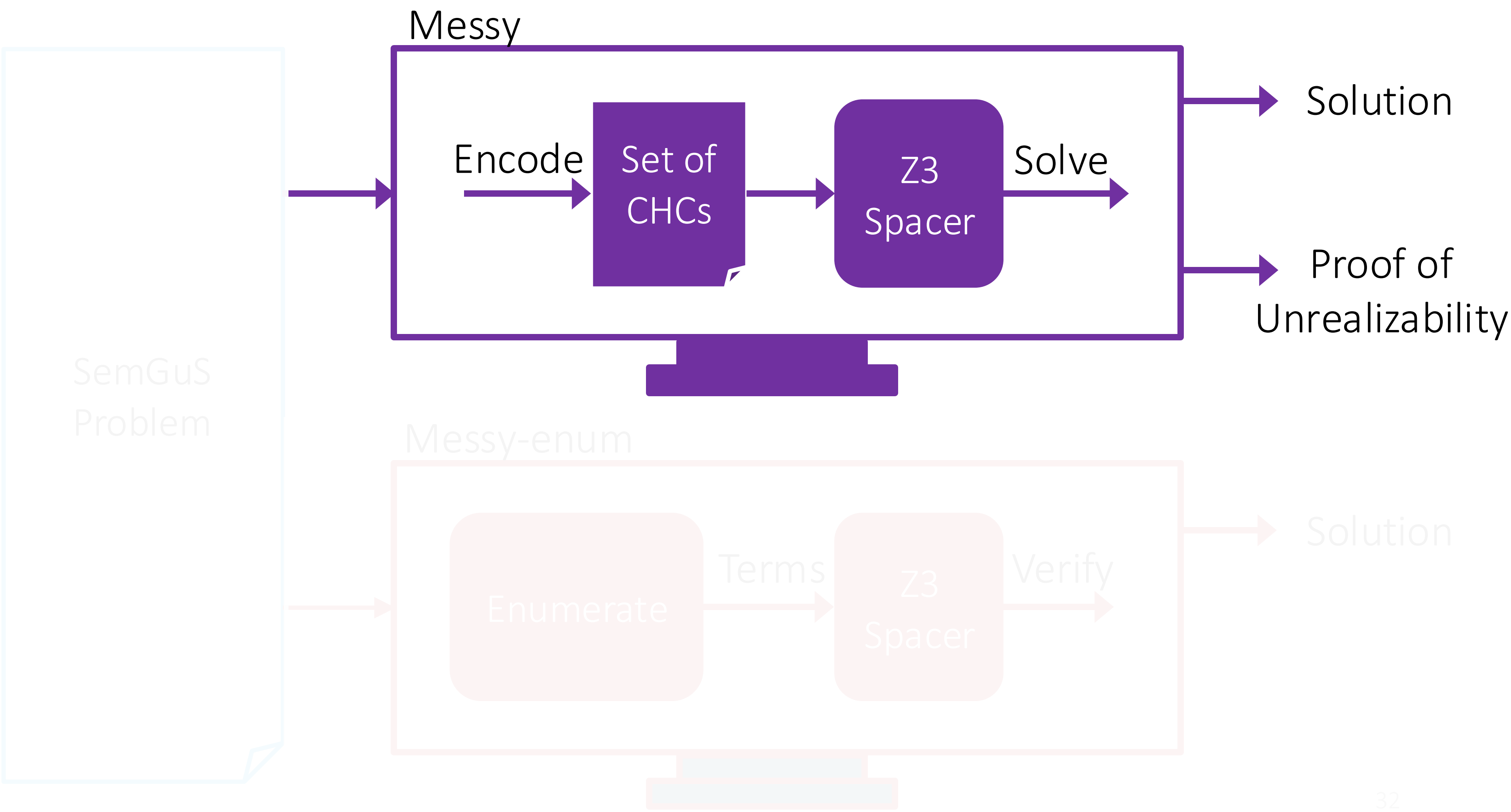
Yes

Logic and Formal Languages

Solving semgus problems



Synthesize
Imperative
Programs
Synthesize
Regexes
Synthesize
SMT
Expressions
SemGuS
Behavioral specification
Logical formula
Search space
Regular Tree
Grammar
+
Constrained Horn
Clauses
??
??
??
??
??
??


Messy
Encode
Set of CHCs
Z3 Spacer
Solve
Solution
Proof of Unrealizability
SemGuS Problem
Messy-enum
Enumerate
Terms
Z3 Spacer
Verify
Solution

SemGuS Problem

Language Syntax

Regular Tree Grammar

$$
\begin{aligned}
&Start ::= While\ B\ do\ S\\
&B ::= E < E\\
&E ::= x \mid y \mid E \,\&\, E \mid (E \mid E)\\
&S ::= S; S \mid x \coloneqq E \mid y \coloneqq E
\end{aligned}
$$

Language Semantics

Constrained Horn Clauses

$$\frac{Sem_E(\Gamma, t_e, v)\quad \Gamma_1 = \Gamma[x \mapsto v]}{Sem_S(\Gamma, x \coloneqq t_e, \Gamma_1)}$$

Behavioral Spec

Constraints / Examples

Examples:
$f(6,9) = (15, \_)$

$$?\ \frac{\boxed{t \in L(Start)}\qquad \boxed{Sem_{Start}}\left(\boxed{(6,9), t, (15, \_)}\right)}{Realizable}$$

Solving a SemGuS problem ⇒ Solving a query over CHCs!

Syntactic constraints as Constrained Horn Clauses

$S ::= S; S$

if $t_1 \in L(S)$ and $t_2 \in L(S)$ then $t_1; t_2 \in L(S)$

$$Syn_S \subseteq Term$$
$$Syn_S(t) \Leftrightarrow t \in L(S)$$

$$\frac{Syn_S(t_1) \quad Syn_S(t_2)}{Syn_S(t_1; t_2)}$$

SemGuS Problem

Language Syntax	Language Semantics	Behavioral Spec
Regular Tree Grammar	Constrained Horn Clauses	Constraints / Examples

$Start ::= While\ B\ do\ S$
$B ::= E < E$
$E ::= x \mid y \mid E \,\&\, E \mid (E \mid E)$
$S ::= S; S \mid x \coloneqq E \mid y \coloneqq E$

$$\frac{Sem_E(\Gamma, t_e, v) \quad \Gamma_1 = \Gamma[x \mapsto v]}{Sem_S(\Gamma, x \coloneqq t_e, \Gamma_1)}$$

Examples:
$\mathrm{f}(6, 9) = (15, \_)$

$$\frac{Syn_{Start}(t) \quad Sem_{Start}((6, 9), t, (15, \_))}{Realizable(t)}$$

Solving a SemGuS problem ⇒ Solving a query over CHCs!

Program Synthesis: Proving $Realizable$

$$\frac{Syn_{Start}(t) \quad Sem_{Start}\big((6,9), t, (15, \_)\big)}{Realizable(t)}$$

Specification: $[\mathrm{f}(6,9) = (15, \_)]$
Valid solution $t = While\ x < y\ do\ x := x \mid y$

Program Synthesis: Proving *Realizable*

$$\frac{Syn_B(x < y) \quad Syn_S(x := x \mid y)}{Syn_{Start}(t)}$$

$$\uparrow$$

$$\frac{Syn_{Start}(t) \quad Sem_{Start}\big((6,9), t, (15, \_)\big)}{Realizable(t)}$$

Specification: $[\mathrm{f}(6,9) = (15, \_)]$
Valid solution $t = While\ x < y\ do\ x := x \mid y$

Program Synthesis: Proving *Realizable*

$$\frac{Syn_B(x < y) \quad Syn_S(x \coloneqq x \mid y)}{Syn_{Start}(t)} \longleftarrow \frac{Syn_B(b) \quad Syn_S(s)}{Syn_{Start}(While\ b\ do\ s)}$$

$$\frac{Syn_{Start}(t) \quad Sem_{Start}((6,9), t, (15, \_))}{Realizable(t)}$$

Specification: $[\mathrm{f}(6,9) = (15, \_)]$

Valid solution $t = While\ x < y\ do\ x \coloneqq x \mid y$

Program Synthesis: Proving *Realizable*

$$\dfrac{\dfrac{Syn_E(x) \quad Syn_E(y)}{Syn_B(x < y)} \qquad \dfrac{\dfrac{Syn_E(x) \quad Syn_E(y)}{Syn_E(x \mid y)}}{Syn_S(x \coloneqq x \mid y)}}{Syn_{Start}(t)}$$

$$\dfrac{Syn_{Start}(t) \quad Sem_{Start}\big((6,9), t, (15, \_)\big)}{Realizable(t)}$$

Specification: $[\mathrm{f}(6,9) = (15, \_)]$

Valid solution $t = While\ x < y\ do\ x \coloneqq x \mid y$

Program Synthesis: Proving *Realizable*

$$\frac{}{Sem_{Start}((6,9), t, (15, \_))}$$

$$\uparrow$$

$$\frac{Syn_{Start}(t) \quad Sem_{Start}((6,9), t, (15, \_))}{Realizable(t)}$$

Specification: $[\mathrm{f}(6,9) = (15, \_)]$
Valid solution $t = While\ x < y\ do\ x := x \mid y$

Program Synthesis: Proving *Realizable*

$$\frac{Sem_B(\Gamma, t_b, v) \quad v = True \quad Sem_S(\Gamma, t_s, \Gamma_1) \quad Sem_{Start}(\Gamma_1, While\ t_b\ do\ t_s, \Gamma_2)}{Sem_{Start}(\Gamma, While\ t_b\ do\ t_s, \Gamma_2)}$$

$$\frac{Sem_B\big((6,9), x < y, True\big) \quad Sem_S\big((6,9), x \coloneqq x \mid y, (15,9)\big) \quad \cdots}{Sem_{Start}\big((6,9), t, (15, \_)\big)}$$

$$\frac{Syn_{Start}(t) \quad Sem_{Start}\big((6,9), t, (15, \_)\big)}{Realizable(t)}$$

Specification: $[f(6,9) = (15, \_)]$

Valid solution $t = While\ x < y\ do\ x \coloneqq x \mid y$

Program Synthesis: Proving *Realizable*

Built a proof tree using t: Program t is a solution!

$$\dfrac{\dfrac{Sem_E\big((6,9),x,6\big)\quad Sem_E\big((6,9),y,9\big)}{Sem_B\big((6,9),x<y,True\big)}\qquad \dfrac{\cdots}{Sem_S\big((6,9),x:=x\mid y,(15,9)\big)}\qquad \cdots}{Sem_{Start}\big((6,9),t,(15,\_)\big)}$$

$$\uparrow$$

$$\dfrac{Syn_{Start}(t)\quad Sem_{Start}\big((6,9),t,(15,\_)\big)}{Realizable(t)}$$

Specification: $[f(6,9) = (15,\_)]$

Valid solution $t = While\ x < y\ do\ x := x \mid y$

The power of programmable frameworks

SemGuS Problem

Language Syntax

Regular Tree Grammar

Weighted Tree Grammar [CAV18]

Language Semantics

Constrained Horn Clauses

Behavioral Spec

Constraints / Examples

SemGuS Problem

Language Syntax

Regular Tree Grammar

Weighted Tree Grammar [CAV18]

Language Semantics

Constrained Horn Clauses

Standard Semantics

Abstract Semantics

[Wang, et al. POPL17]

Bounded Loops

Sketch/Rosette

Custom Ops

DSL-based solvers

Behavioral Spec

Constraints / Examples

Abstract semantics

$x \,\&\, y$

$$\frac{[\![\, x \,]\!](\Gamma) = v_x \quad [\![\, y \,]\!](\Gamma) = v_y \quad v = v_x \,\&\, v_y}{[\![\, x\&y \,]\!](\Gamma) = v}$$

Abstract Domain: Consider only first bit

no solution using abstract semantics

$\Rightarrow$

no solution using standard semantics

$$\frac{Sem_E(\Gamma, x, b_x) \quad Sem_E(\Gamma, y, b_y) \quad b = if\ (b_x = \top \vee b_y = \top)\ \top\ else\ b_x \,\&\, b_y}{Sem_S(\Gamma, x\&y, b)}$$

Underapproximated semantics: Bounded Loops

$While\ b\ do\ s$

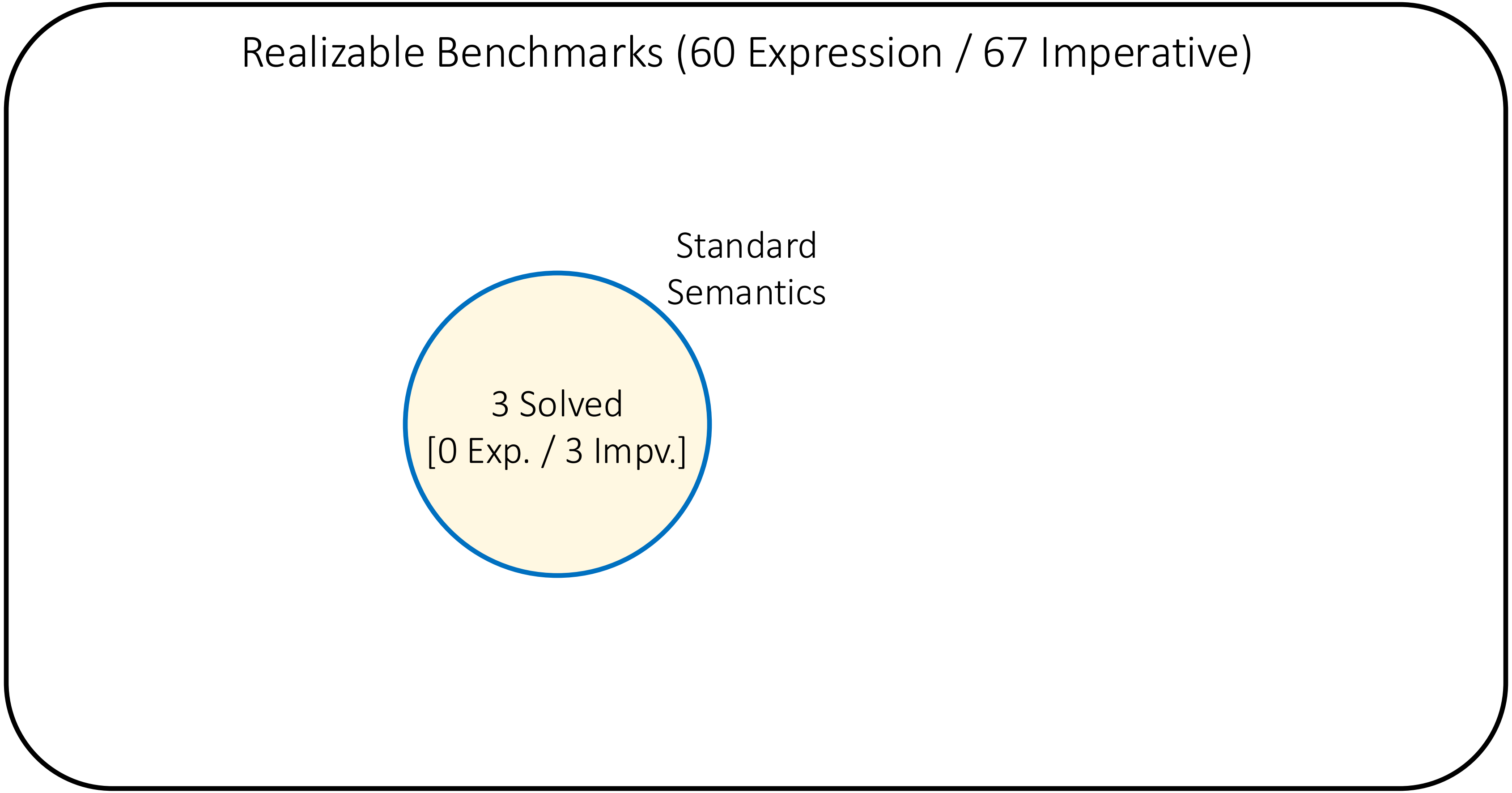
$$\frac{[\![\, b \,]\!](\Gamma) = True \quad [\![\, s \,]\!](\Gamma) = \Gamma_1 \quad [\![\, While\ b\ do\ s \,]\!](\Gamma_1) = \Gamma_2}{[\![\, While\ b\ do\ s \,]\!](\Gamma) = \Gamma_2}$$

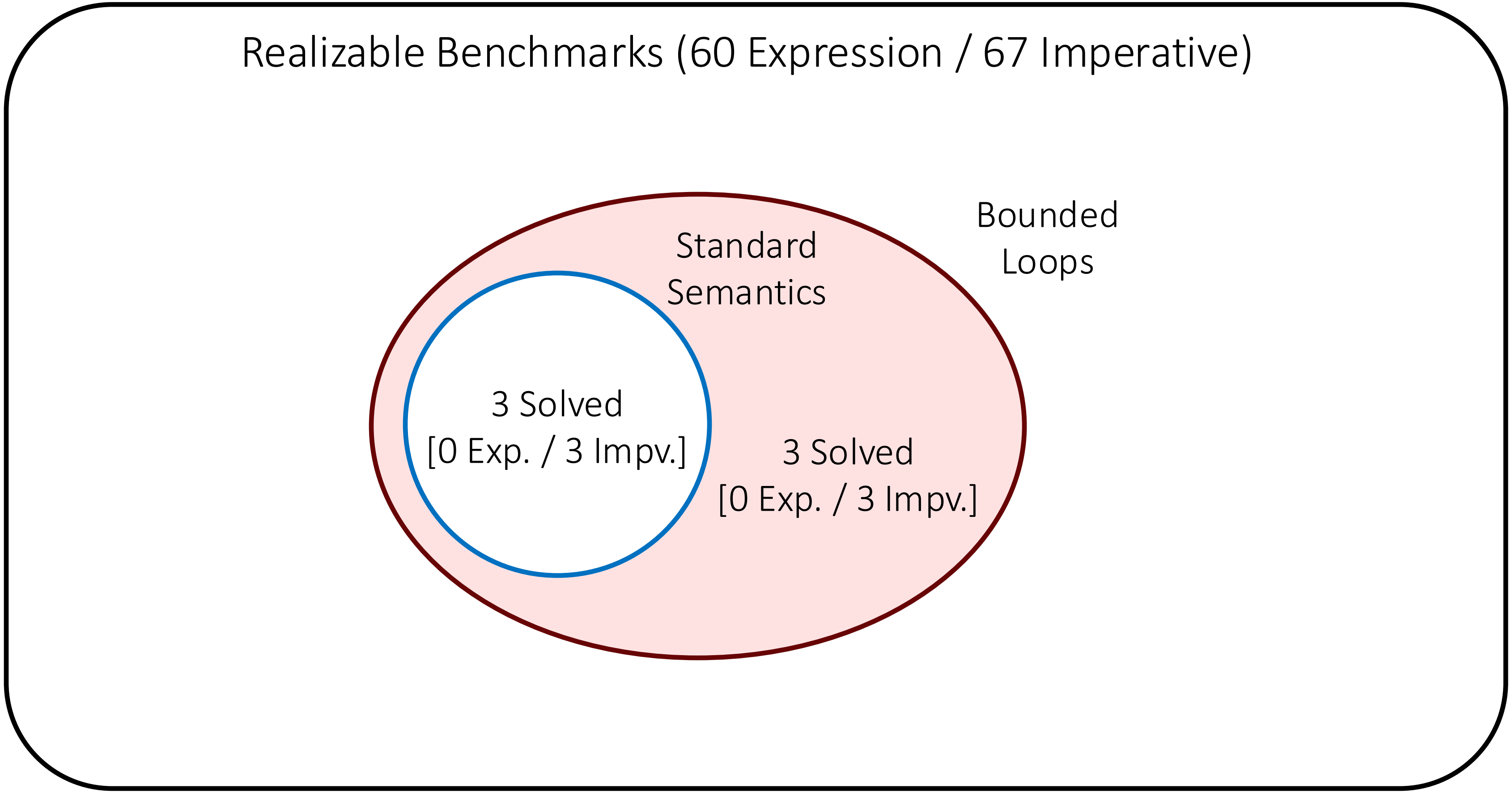
Bound Loops to n iterations

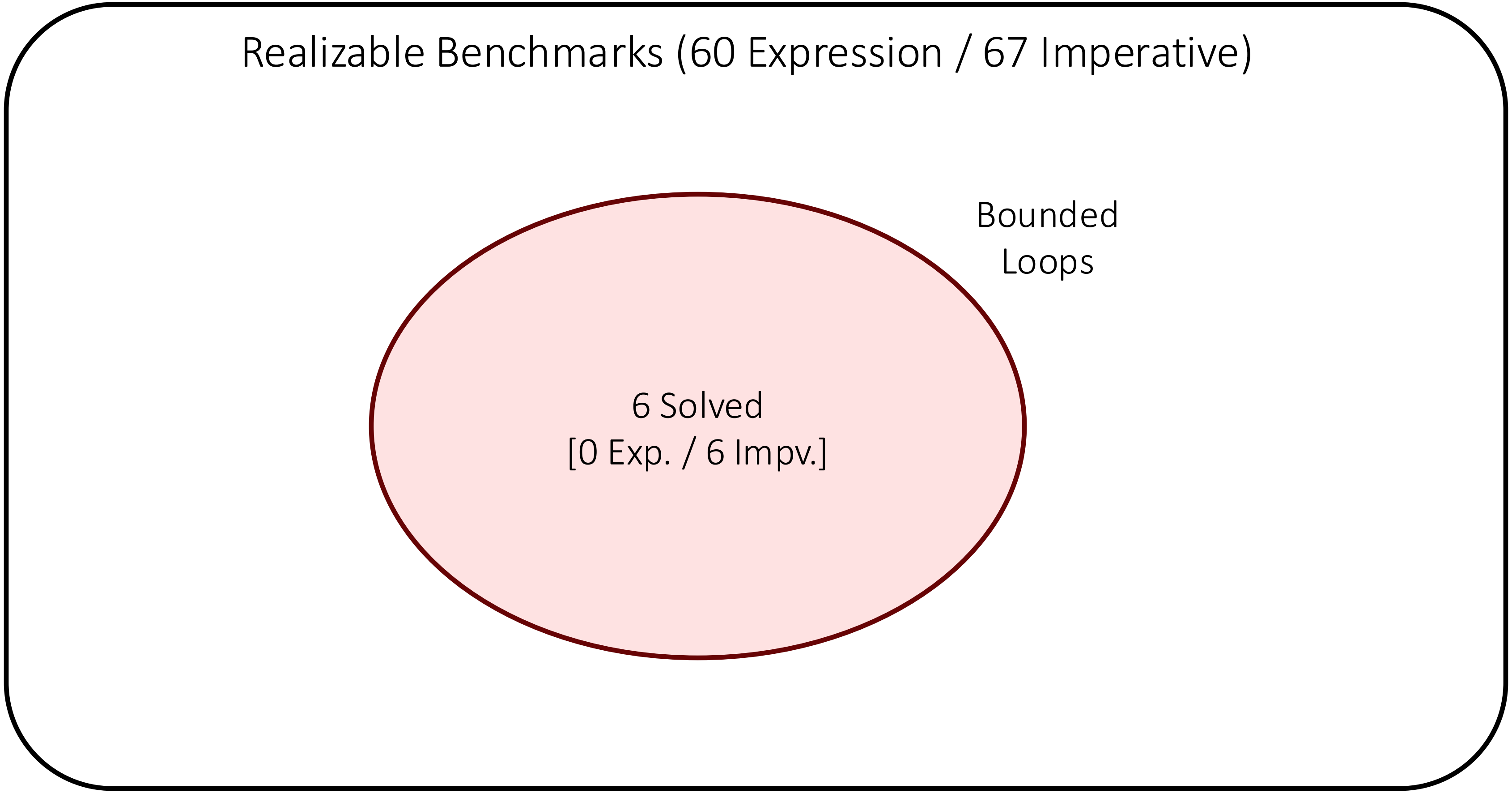
Solution using bounded loops ⇒ solution using standard semantics

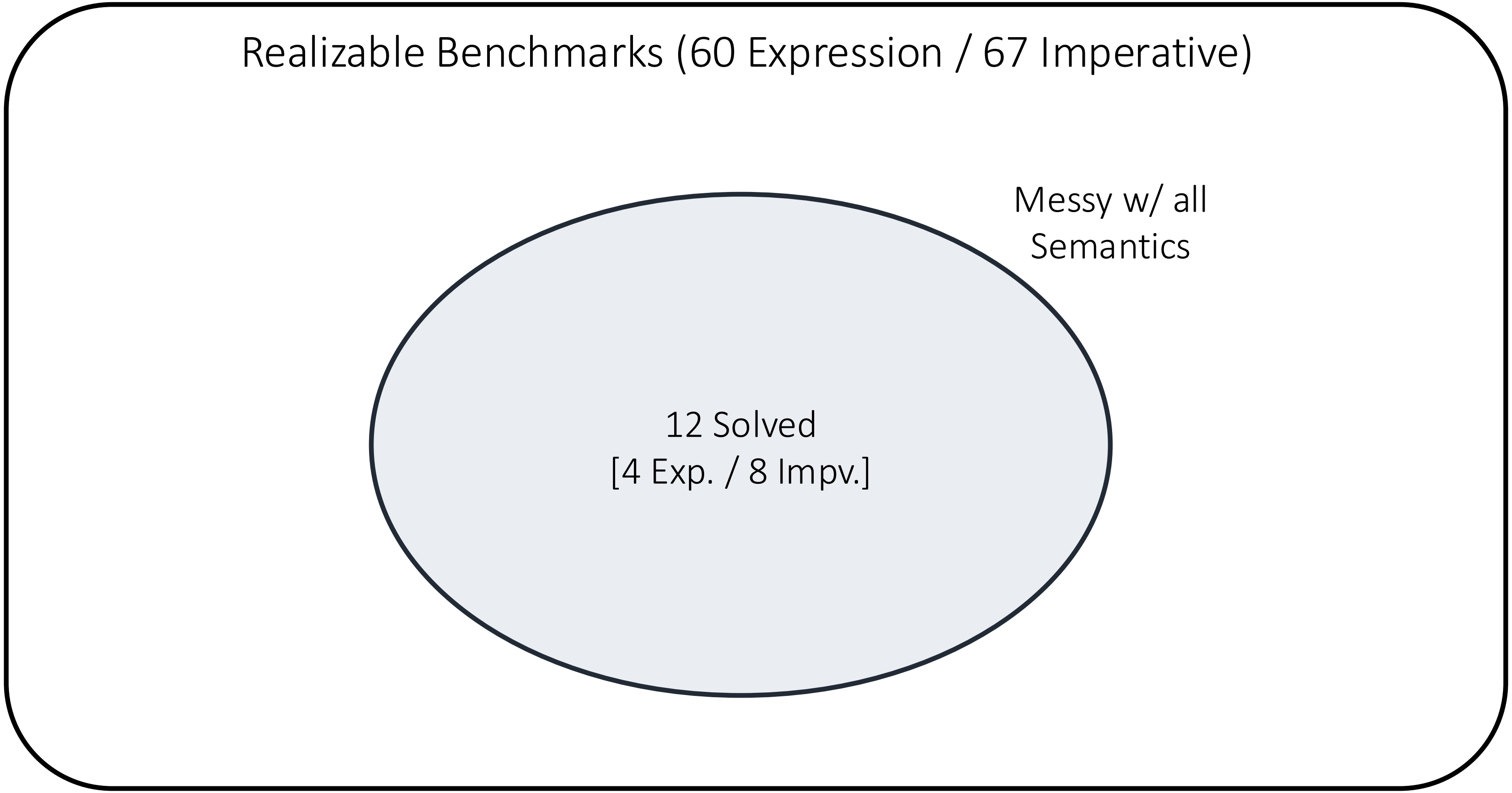
$$\frac{Sem_B(\Gamma, b, True) \quad k > 0 \quad Sem_S(\Gamma, s, \Gamma_1) \quad Sem_S(\Gamma_1, While\ b\ do\ s, \Gamma_2, k-1)}{Sem_S(\Gamma, While\ b\ do\ s, \Gamma_2, k)}$$

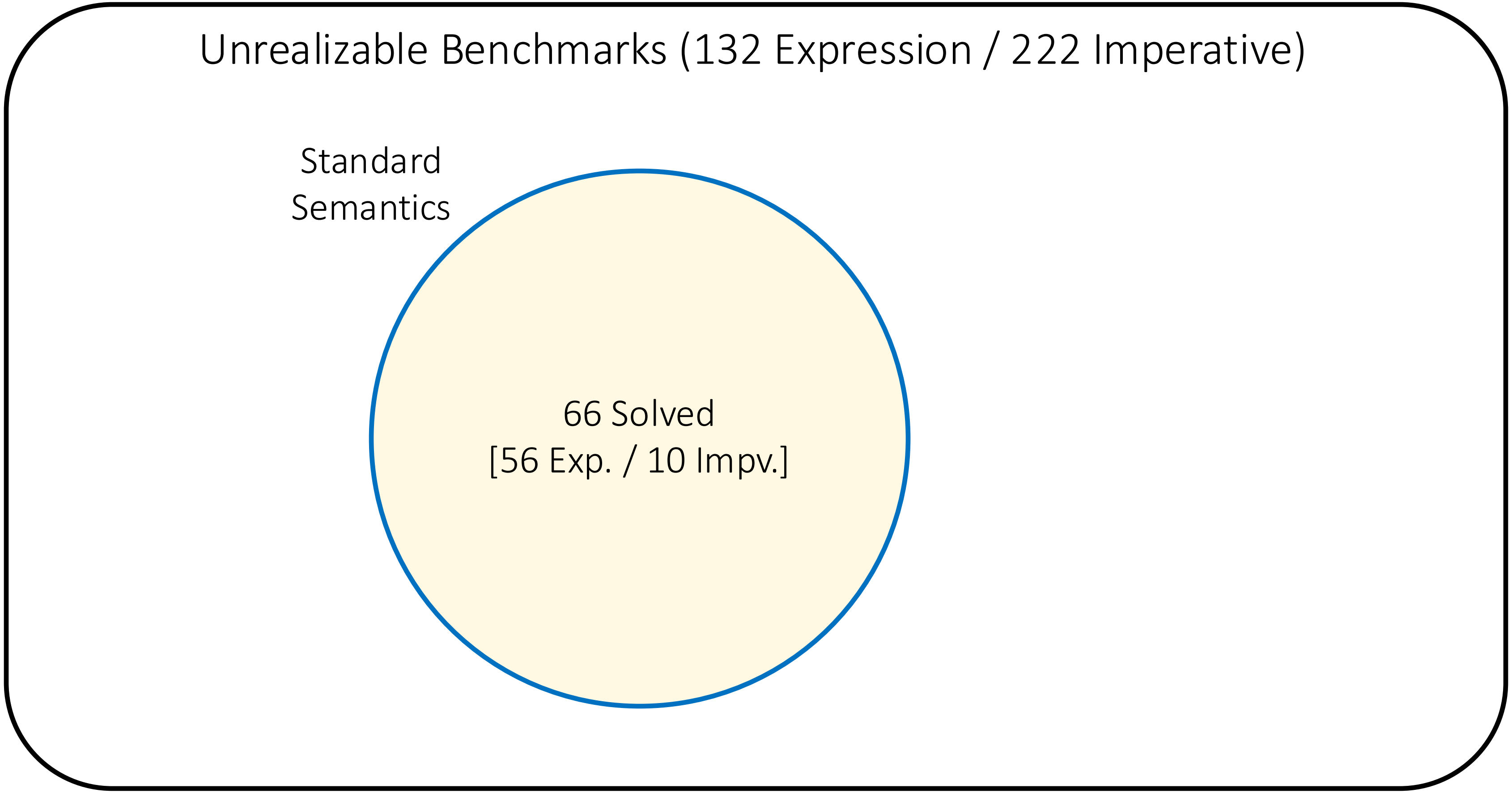
Semgus in practice


Realizable Benchmarks (60 Expression / 67 Imperative)
Standard
Semantics
3 Solved
[0 Exp. / 3 Impv.]

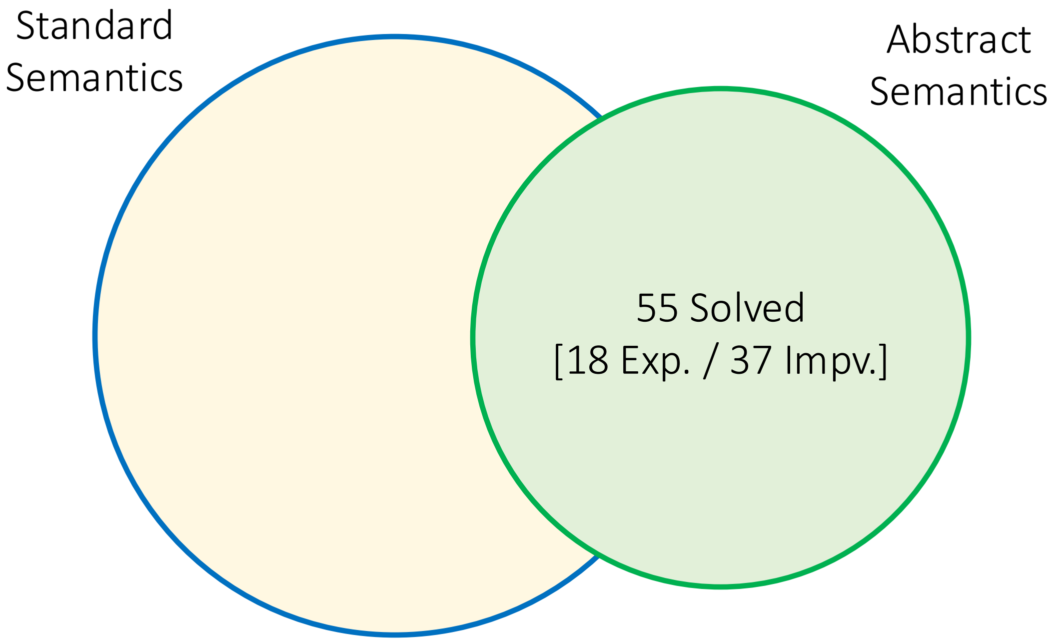

Realizable Benchmarks (60 Expression / 67 Imperative)
Bounded Loops
Standard Semantics
3 Solved [0 Exp. / 3 Impv.]
3 Solved [0 Exp. / 3 Impv.]


Realizable Benchmarks (60 Expression / 67 Imperative)
Bounded
Loops
6 Solved
[0 Exp. / 6 Impv.]

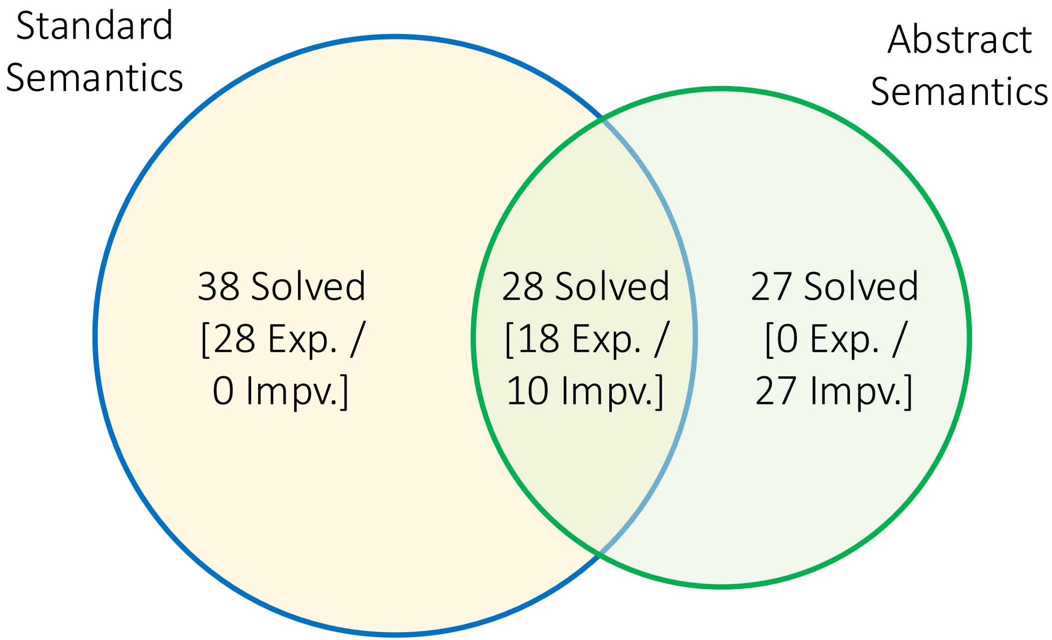

Realizable Benchmarks (60 Expression / 67 Imperative)
Messy w/ all Semantics
12 Solved
[4 Exp. / 8 Impv.]

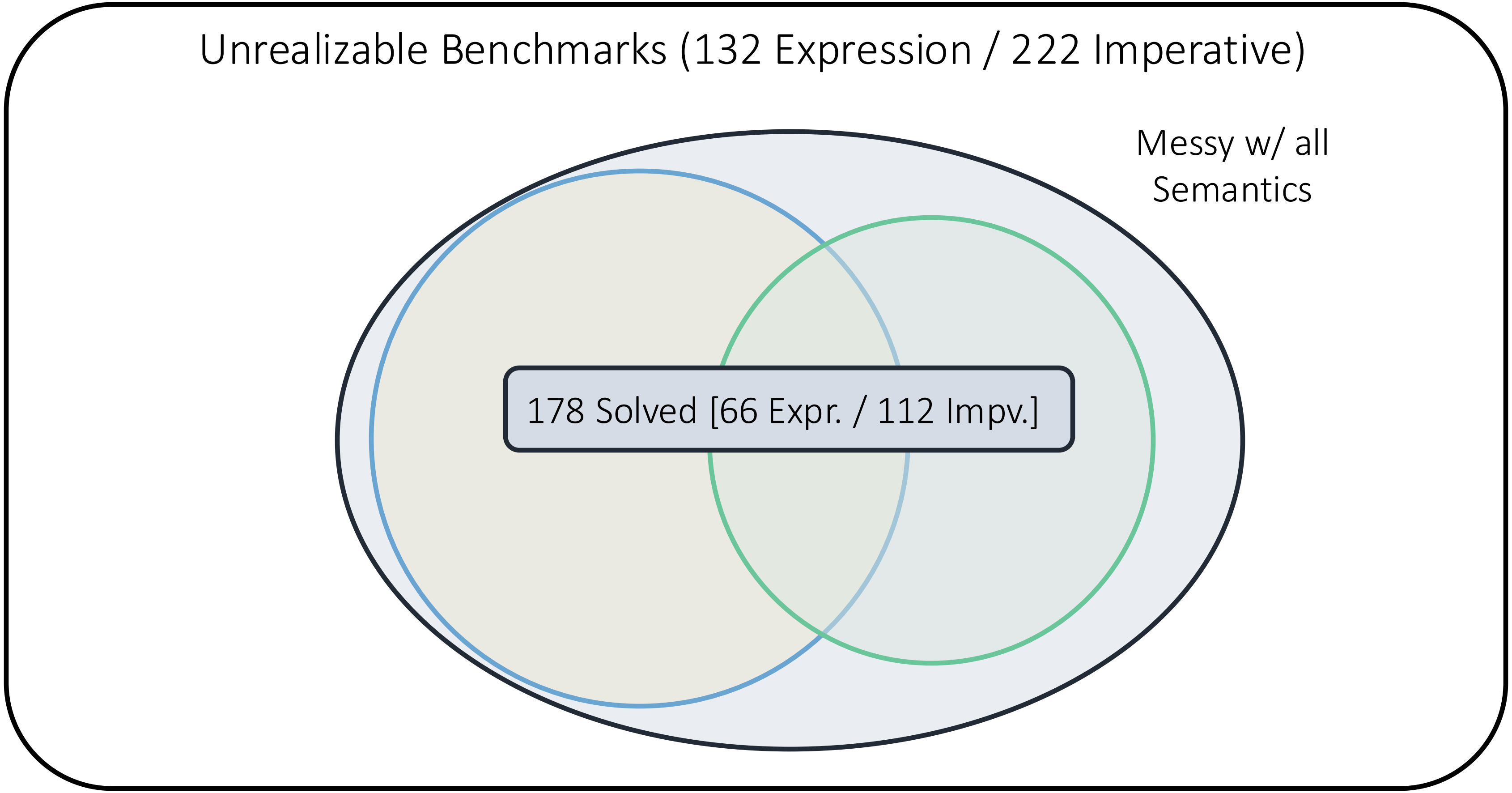

Unrealizable Benchmarks (132 Expression / 222 Imperative)
Standard
Semantics
66 Solved
[56 Exp. / 10 Impv.]

Unrealizable Benchmarks (132 Expression / 222 Imperative)

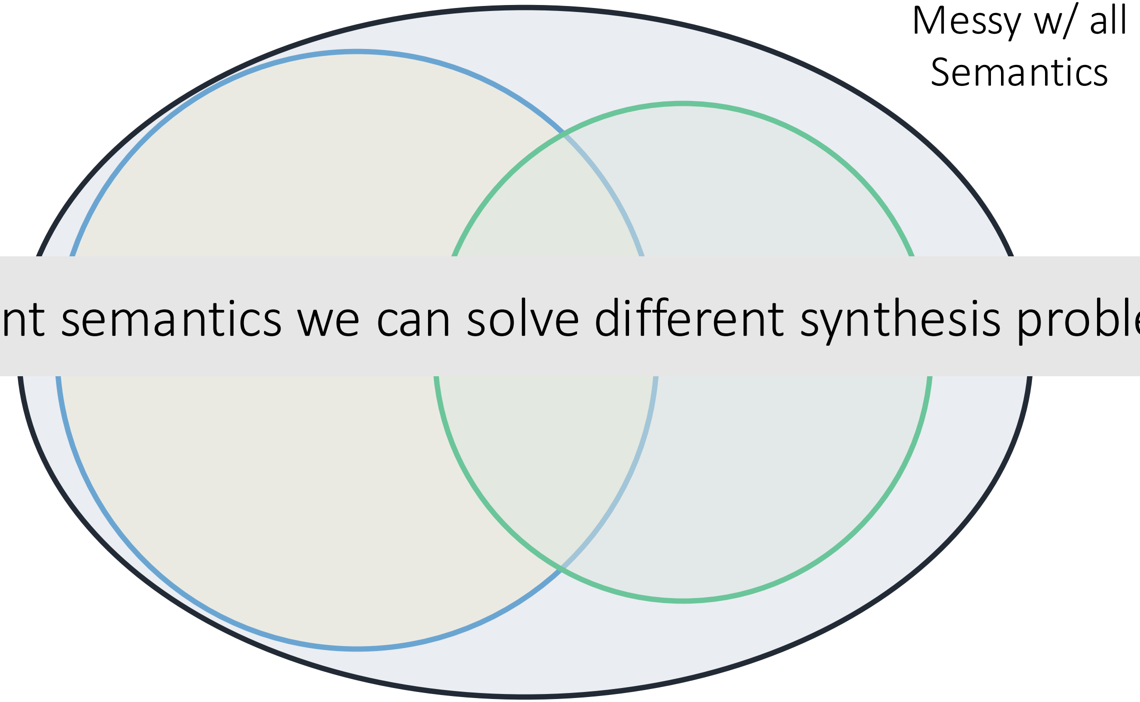


Unrealizable Benchmarks (132 Expression / 222 Imperative)




Unrealizable Benchmarks (132 Expression / 222 Imperative)
Messy w/ all Semantics
178 Solved [66 Expr. / 112 Impv.]

Unrealizable Benchmarks (132 Expression / 222 Imperative)



With different semantics we can solve different synthesis problems!

Syntax:

$Start ::= While\ B\ do\ S$

$B ::= E < E$

$E ::= \boxed{x \mid y \mid E \,\&\, E \mid (E \mid E)}$

$S ::= S; S \mid x := E \mid y := E$

$\mathrm{Sem}_{\mathrm{Start}}$ satisfies the following lemma:

$\forall t, x', y'.\ Sem_{Start}\big((14, 15), t, (x', y')\big) \Rightarrow x' \,\&\, 4 = 4$

For the desired output, $1 \,\&\, 4 \neq 4$: Problem is unrealizable!

$$\frac{Syn_{Start}(t) \quad Sem_{Start}\big((6, 9), t, (15, \_)\big) \quad \boxed{Sem_{Start}\big((14, 15), t, (1, \_)\big)}}{Realizable}$$

Specification: $[\mathrm{f}(6, 9) = (15, \_), \mathrm{f}(14, 15) = (1, \_)]$

What’s so good about unrealizability?

How it all started

We found out <u>synthesizers were unpredictable</u> [PLDI17]

Synthesis with <u>quantitative objectives</u>

- Syntactic and semantic objectives [CAV16, ESOP20, CAV18, SAS19]
- Probabilistic objectives [IJCAI17, OOPSLA17, CAV17, CAV19]
- Resource bounds [CAV21]

To synthesize the optimal programs we have to prove <u>unrealizability</u>

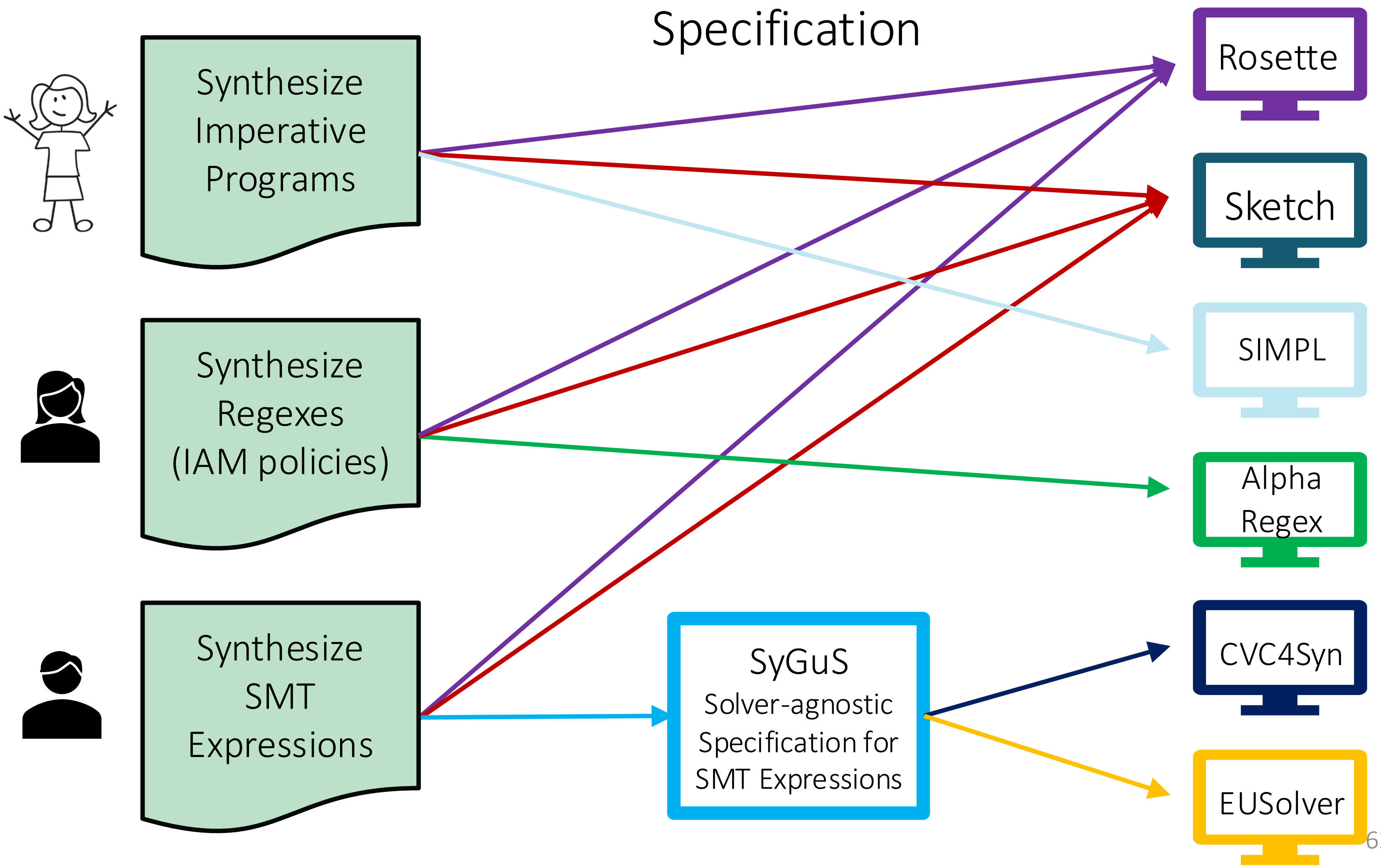
- SyGuS unrealizability [CAV19, PLDI20a]
- Imperative programs and beyond [POPL21]

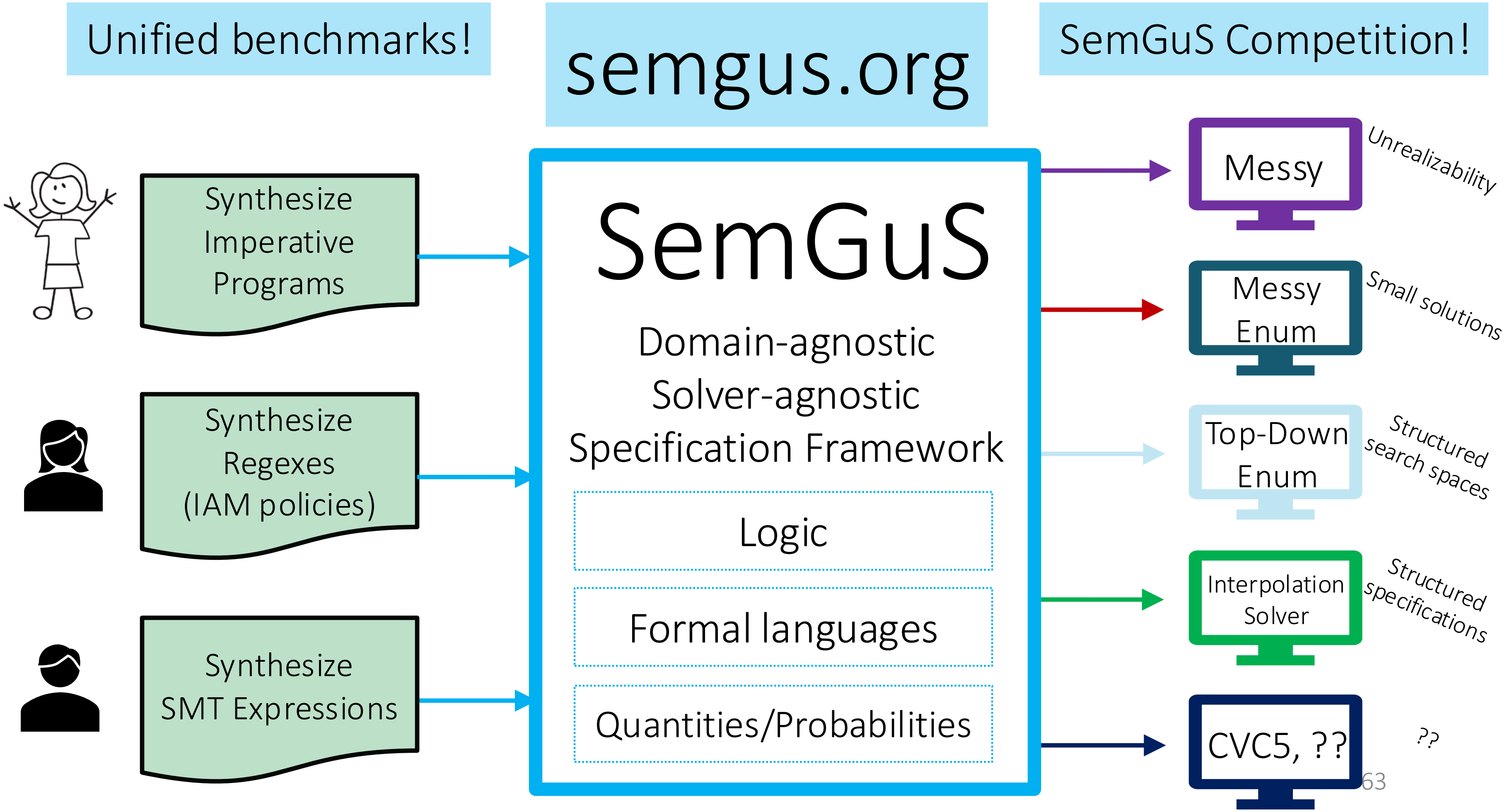
Applied ideas to <u>many domains</u>

- Networks [POPL17, SIGMETRICS18, PLDI20b, SIGCOMM21]
- Program and data transformations [FSE17, ICSE17, OOPSLA19]

SemGuS

Conclusion


Specification
Synthesize Imperative Programs
Synthesize Regexes (IAM policies)
Synthesize SMT Expressions
SyGuS
Solver-agnostic Specification for SMT Expressions
Rosette
Sketch
SIMPL
Alpha Regex
CVC4Syn
EUSolver


Unified benchmarks!
semgus.org
SemGuS Competition!
Synthesize Imperative Programs
Synthesize Regexes (IAM policies)
Synthesize SMT Expressions
SemGuS
Domain-agnostic
Solver-agnostic
Specification Framework
Logic
Formal languages
Quantities/Probabilities
Messy
Unrealizability
Messy Enum
Small solutions
Top-Down Enum
Structured search spaces
Interpolation Solver
Structured specifications
CVC5, ??
??

SemGuS: contributions

Unbounded correctness guarantees for arbitrary programs

CHC to express semantics of programs in a way that is agnostic of the specific domain or DSL

Natural encoding as CHC solving

First to prove unrealizability for complex problems

Semantics can be changed to help solvers

SemGuS : limitations

CHCs are not trivial to write

Very limited scalability

SemGuS : questions

Behavioral constraints? Structural constraints? Search strategy?

- Examples
- Grammar + Semantics
- Constraint-based using CHC + different encodings

What happens when we combine semantics?

$$\frac{[\![b]\!](\Gamma, v_b) \qquad [\![s]\!](\Gamma, \Gamma_1) \qquad [\![while\ b\ do\ s]\!](\Gamma_1, \Gamma_2) \qquad \Gamma_3 = ITE(v_b, \Gamma_2, \Gamma)}{[\![while\ b\ do\ s]\!](\Gamma, \Gamma_3)}$$

Unnecessarily restrictive!

- requires loop to keep running even if the guard is false!

Finite search spaces

Assume that a SemGuS grammar accepts only finitely many programs. Will Messy still require CHCs to solve the synthesis problem, or can it use symbolic execution techniques like Sketch to build a finite size SMT constraint?

- Still have to deal with loops!

Finite input spaces

If the input space to the problem is finite (e.g., bitvectors), is there a way to solve SemGuS problems without using CHC solvers?

- Grammar flow analysis!

- Can use observational equivalence as there will only be finitely many equivalence classes over program space

Can we verify a program against a spec?

How to implement Messy Enum?

$$\frac{Syn_{Start}(t) \quad Sem_{Start}\big((6,9), t, (15, \_)\big)}{Realizable(t)}$$

$$\frac{Sem_{Start}(x, t, y) \quad not(phi(x, y))}{Counterexample(x)}$$