Unrealizability Logic

Jinwoo Kim, Loris D'Antoni, Thomas Reps

Program Synthesis in a Nutshell

Search Space

Behavioral Spec

Program Synthesis in a Nutshell

Search Space

$S \rightarrow x := BV \mid y := BV \mid S; S$
$BV \rightarrow x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid !\, BV$

Behavioral Spec

Program Synthesis in a Nutshell

Search Space

$S \rightarrow x := BV \mid y := BV \mid S; S$
$BV \rightarrow x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid !\, BV$

BVIMP

Behavioral Spec

Program Synthesis in a Nutshell

Search Space

$S \rightarrow x := BV \mid y := BV \mid S; S$
$BV \rightarrow x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid !\, BV$

BVIMP

Behavioral Spec

$f(x, y) = (x \oplus y, y)$

$f(1, 10) = (11, 10) \wedge$
$f(14{,}15) = (1{,}15)$

Program Synthesis in a Nutshell

Search Space

$S \rightarrow x := BV \mid y := BV \mid S; S$
$BV \rightarrow x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid !\, BV$

BVIMP

Behavioral Spec

$f(x, y) = (x \oplus y, y)$

$f(1, 10) = (11, 10) \wedge$
$f(14{,}15) = (1{,}15)$

BVSPEC

Program Synthesis in a Nutshell

Search Space

$S \rightarrow x := BV \mid y := BV \mid S; S$
$BV \rightarrow x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid !\, BV$

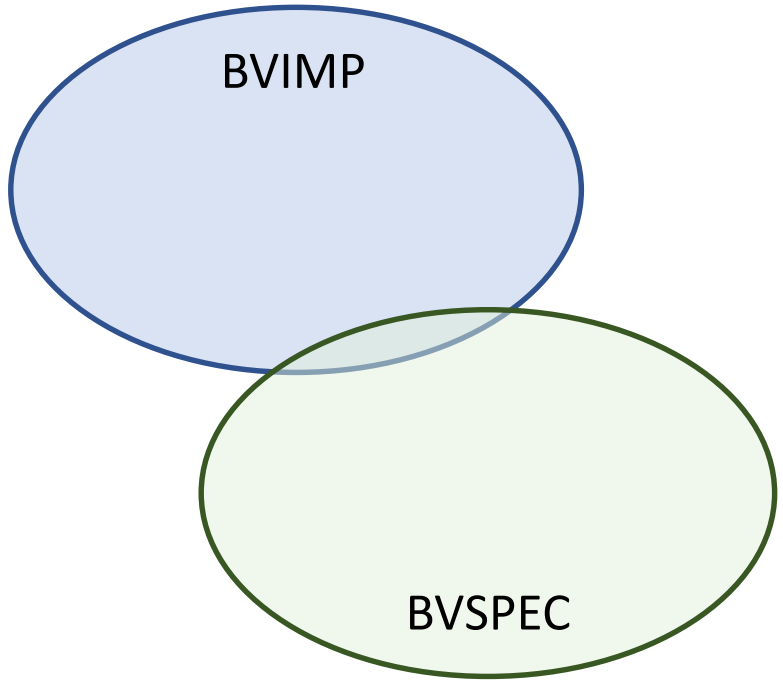
BVIMP

Behavioral Spec

$f(x, y) = (x \oplus y, y)$

$f(1, 10) = (11, 10) \wedge$
$f(14{,}15) = (1{,}15)$

BVSPEC



Program Synthesis in a Nutshell

Search Space

$S \to x := BV \mid y := BV \mid S; S$
$BV \to x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid !\, BV$

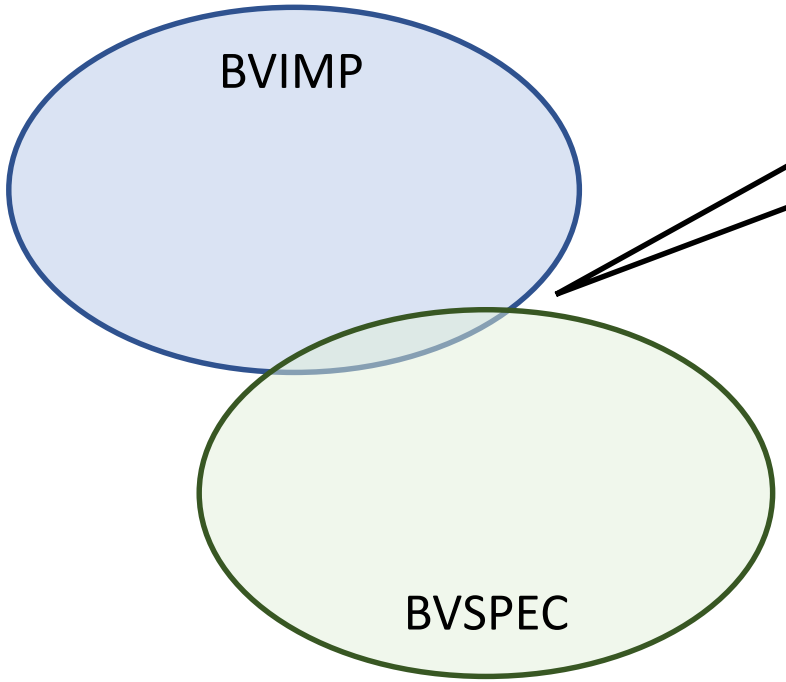
BVIMP

Behavioral Spec

$f(x, y) = (x \oplus y, y)$

$f(1, 10) = (11, 10) \wedge$
$f(14{,}15) = (1{,}15)$

BVSPEC



Goal: Find f in the intersection

Program Synthesis in a Nutshell

Search Space

$S \to x := BV \mid y := BV \mid S; S$
$BV \to x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid !\, BV$

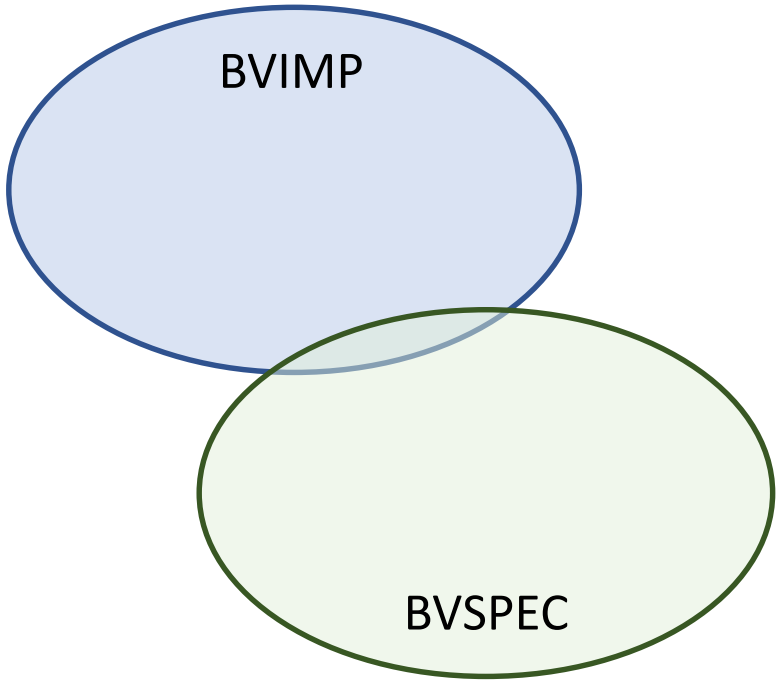
BVIMP

Behavioral Spec

$f(x, y) = (x \oplus y, y)$

$f(1, 10) = (11, 10) \wedge$
$f(14{,}15) = (1{,}15)$

BVSPEC



Goal: Find f in the intersection

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

Sometimes, there is no such f: Unrealizability

Search Space

$S \to x := BV \mid y := BV \mid S; S$
$BV \to x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid \sout{!\, BV}$

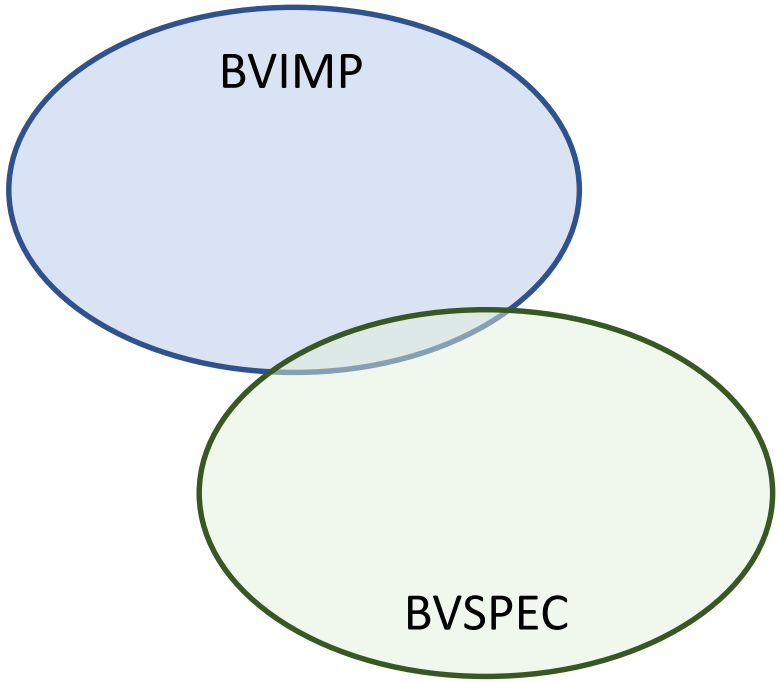
BVIMPN

Behavioral Spec

$f(x, y) = (x \oplus y, y)$

$f(1, 10) = (11, 10) \wedge$
$f(14{,}15) = (1{,}15)$

BVSPEC



Goal: Find f in the intersection

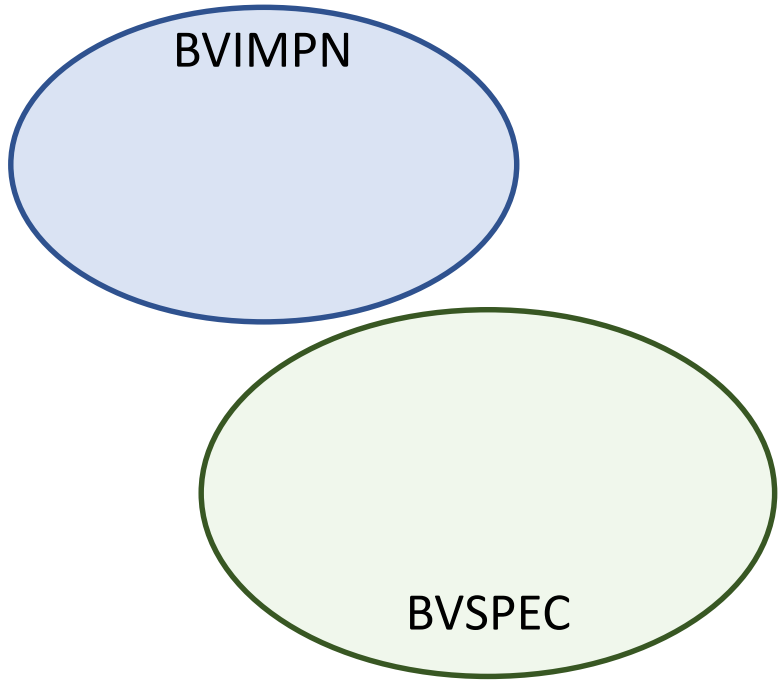
$x := (x \,\&\, !\, y) \mid (!\, x \,\&\, y)$

Sometimes, there is no such f: Unrealizability

Search Space

$S \to x := BV \mid y := BV \mid S; S$
$BV \to x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid$ ~~$!\, BV$~~

BVIMPN



Behavioral Spec

$f(x, y) = (x \oplus y, y)$

$f(1, 10) = (11, 10) \wedge$
$f(14{,}15) = (1{,}15)$

BVSPEC

Goal: Find f in the intersection

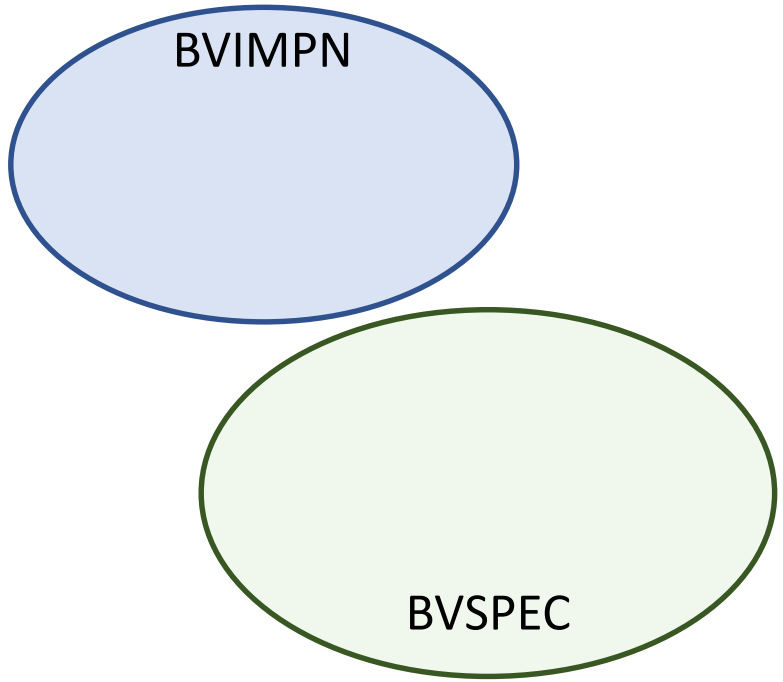
$x := (x \,\&\, !\, y) \mid (!\, x \,\&\, y)$

Sometimes, there is no such f: Unrealizability

Search Space

$S \to x := BV \mid y := BV \mid S; S$
$BV \to x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid$ ~~$!\, BV$~~

BVIMPN



Behavioral Spec

$f(x, y) = (x \oplus y, y)$

$f(1, 10) = (11, 10) \wedge$
$f(14{,}15) = (1{,}15)$

BVSPEC

Goal: Find f in the intersection

~~$x := (x \,\&\, !\, y) \mid (!\, x \,\&\, y)$~~

Now: No such f in the intersection! (Unrealizability)

Sometimes, there is no such f: Unrealizability

Search Space

Goal: Find f in

Paper Goal:
Develop a general proof system for proving that
synthesis problems are unrealizable

BVSPEC

$f(1,10) = (11,10) \wedge$
$f(14,15) = (1,15)$

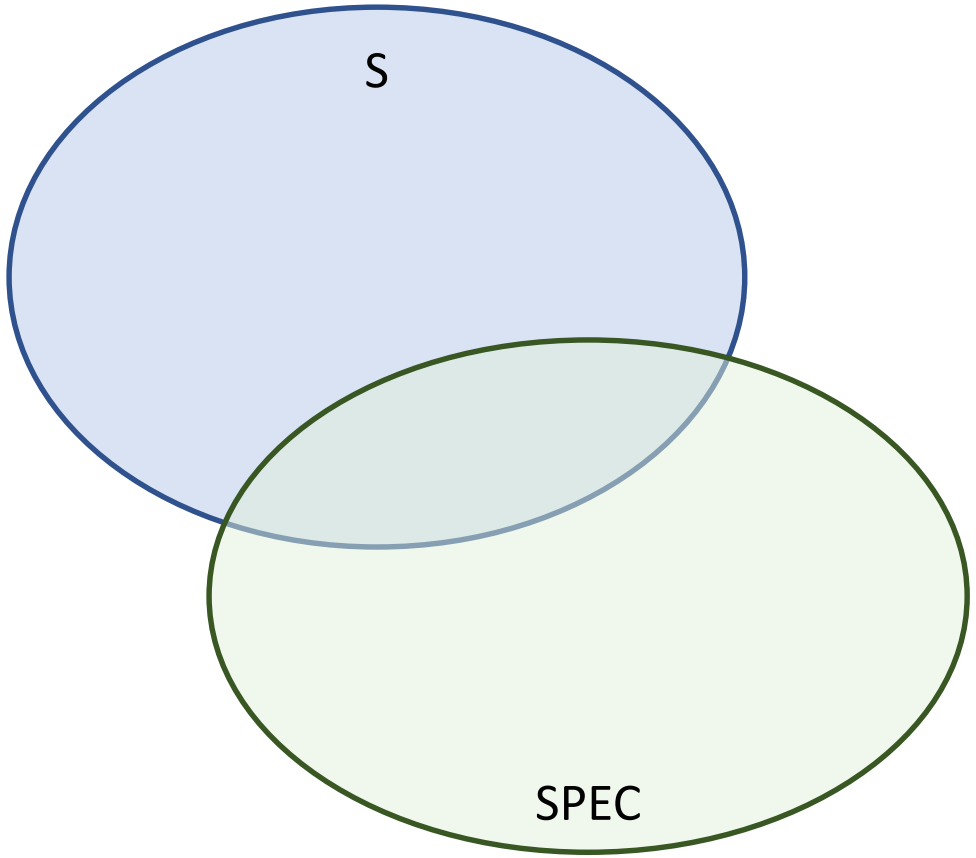
BVSPEC

(Unrealizability)

Motivations for Unrealizability: Optimization

Search Space
S

Behavioral Spec
SPEC

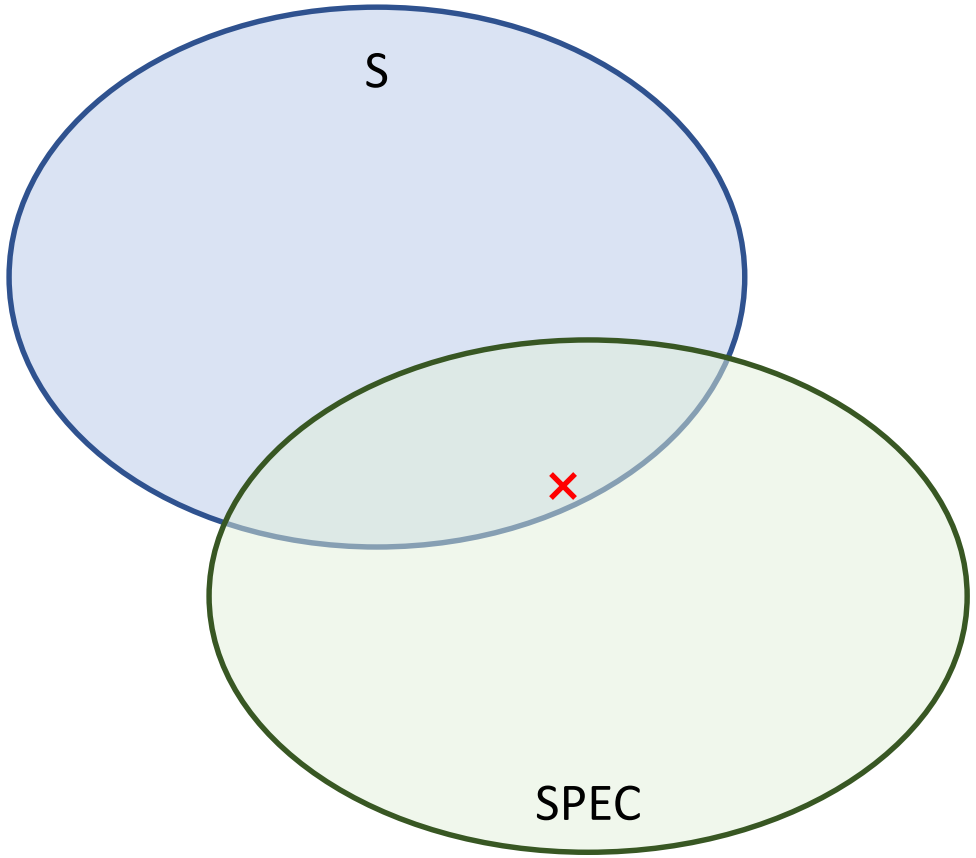


Hu et al. Syntax-Guided Synthesis with Quantitative Syntactic Objectives, CAV '18

Motivations for Unrealizability: Optimization

Search Space
S

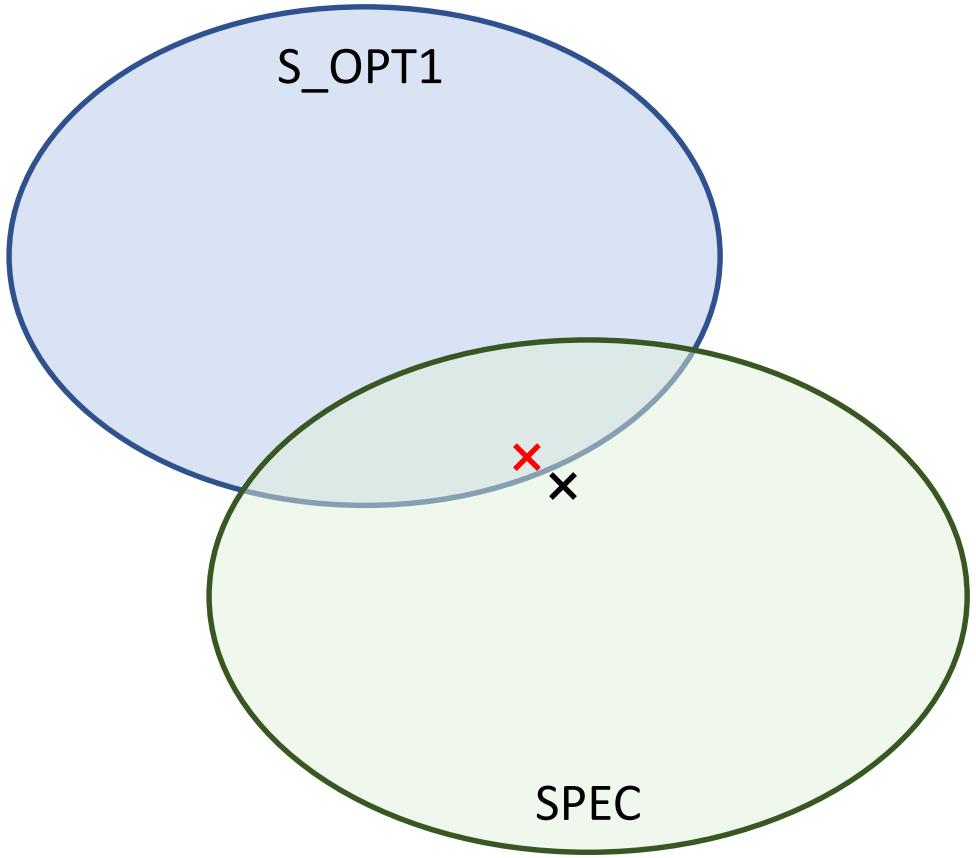
Behavioral Spec
SPEC



Hu et al. Syntax-Guided Synthesis with Quantitative Syntactic Objectives, CAV '18

Motivations for Unrealizability: Optimization

Search Space
S
S_OPT1



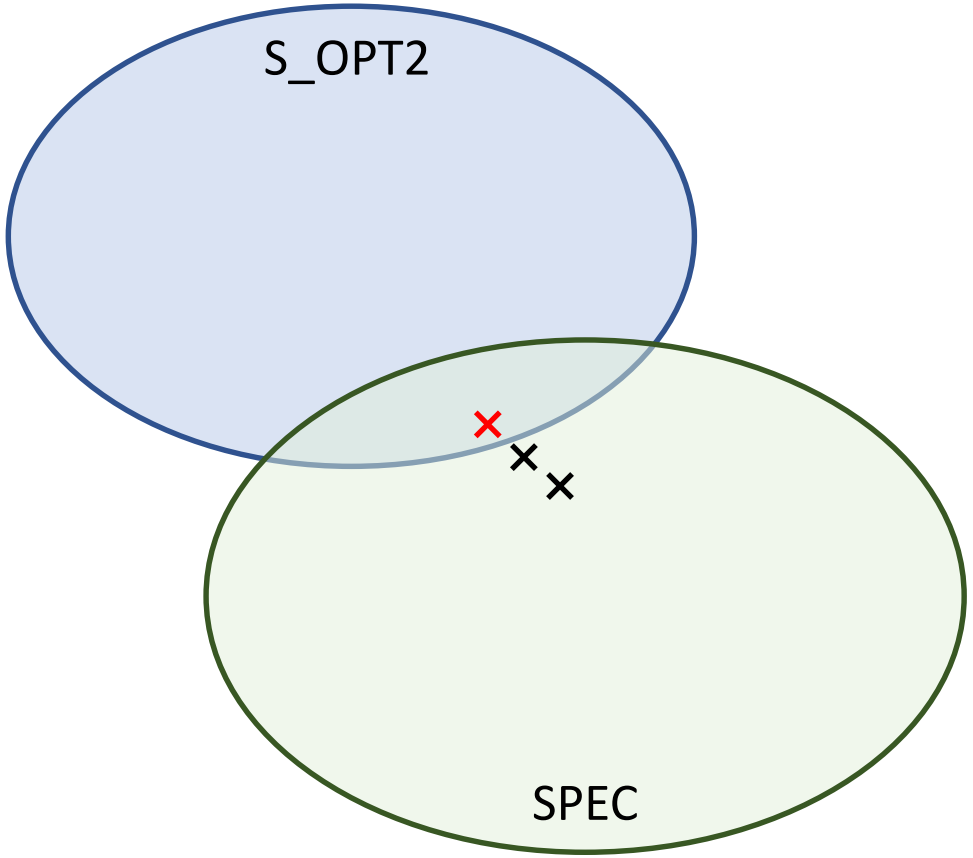
Behavioral Spec
SPEC

Hu et al. Syntax-Guided Synthesis with Quantitative Syntactic Objectives, CAV '18

Motivations for Unrealizability: Optimization

Search Space
S
S_OPT1
S_OPT2

Behavioral Spec
SPEC

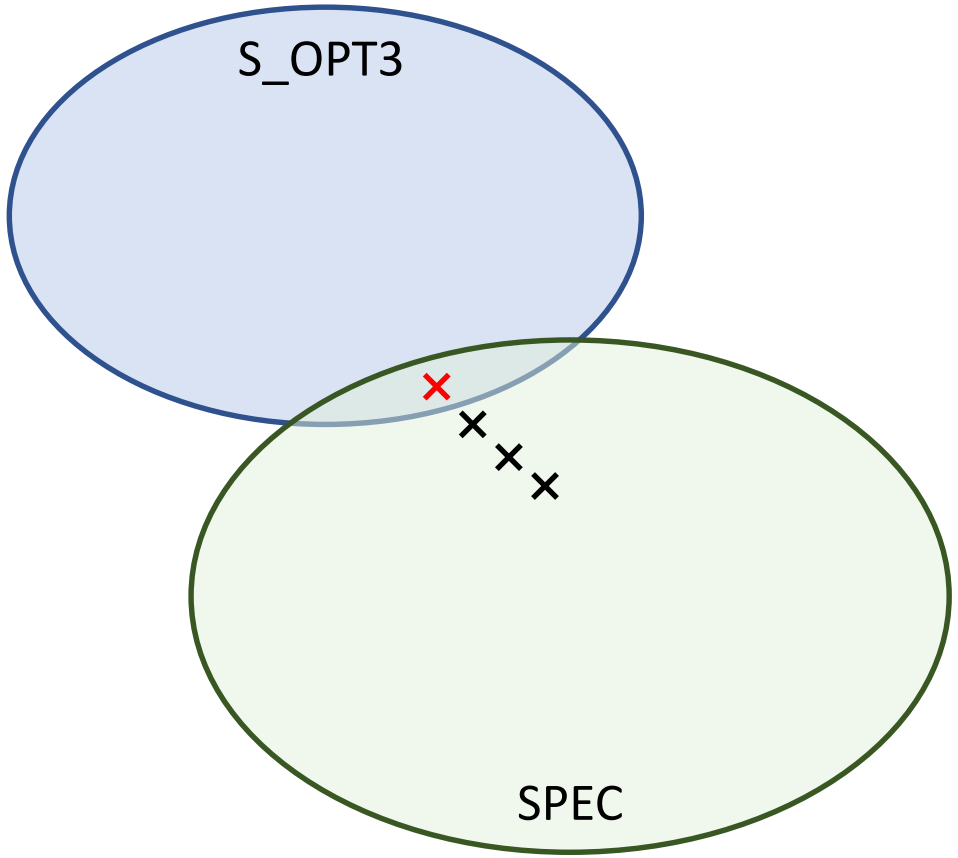


Hu et al. Syntax-Guided Synthesis with Quantitative Syntactic Objectives, CAV '18

Motivations for Unrealizability: Optimization

Search Space
S
S_OPT1
S_OPT2
S_OPT3

Behavioral Spec
SPEC

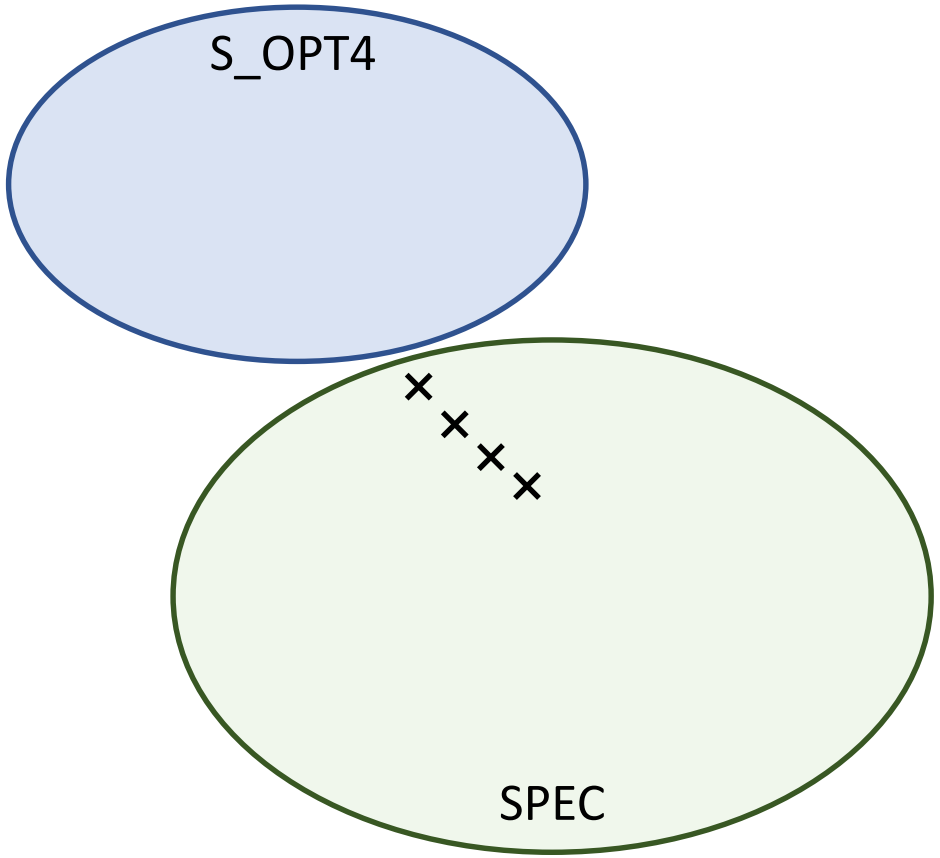


Hu et al. Syntax-Guided Synthesis with Quantitative Syntactic Objectives, CAV '18

Motivations for Unrealizability: Optimization

Search Space
S
S_OPT1
S_OPT2
S_OPT3
S_OPT4

Behavioral Spec
SPEC

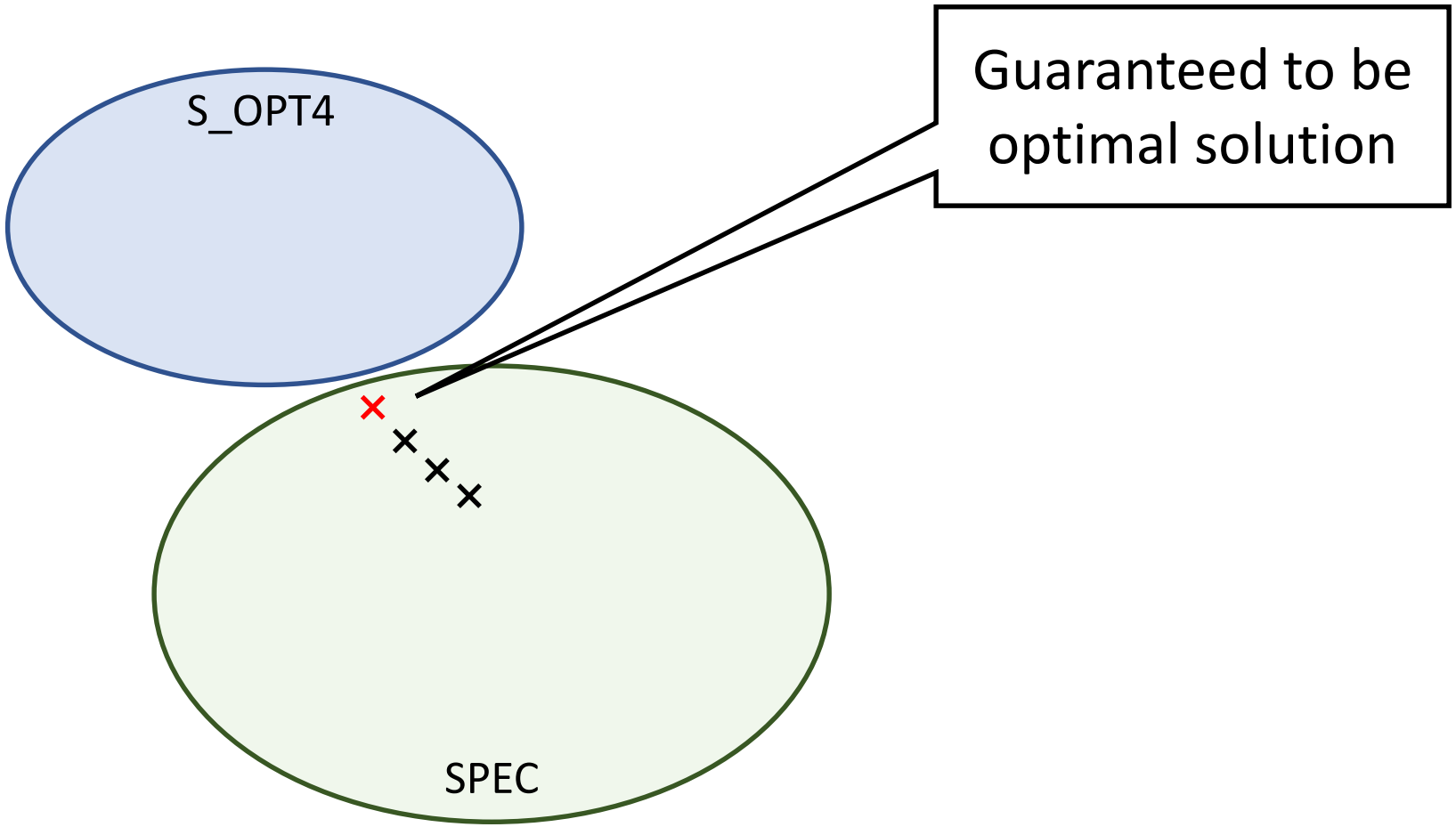


Hu et al. Syntax-Guided Synthesis with Quantitative Syntactic Objectives, CAV '18

Motivations for Unrealizability: Optimization

Search Space
S
S_OPT1
S_OPT2
S_OPT3
S_OPT4

Behavioral Spec
SPEC

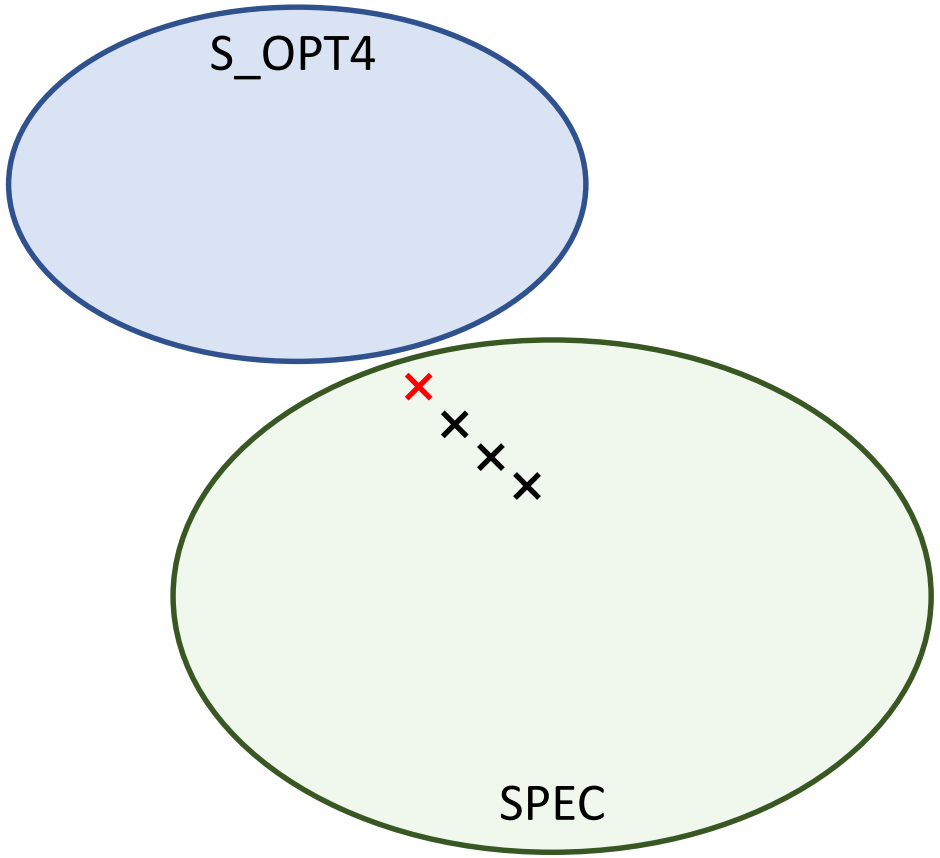


Hu et al. Syntax-Guided Synthesis with Quantitative Syntactic Objectives, CAV '18

Motivations for Unrealizability: Optimization

Search Space
S
S_OPT1
S_OPT2
S_OPT3
S_OPT4

Behavioral Spec
SPEC



Guaranteed to be optimal solution

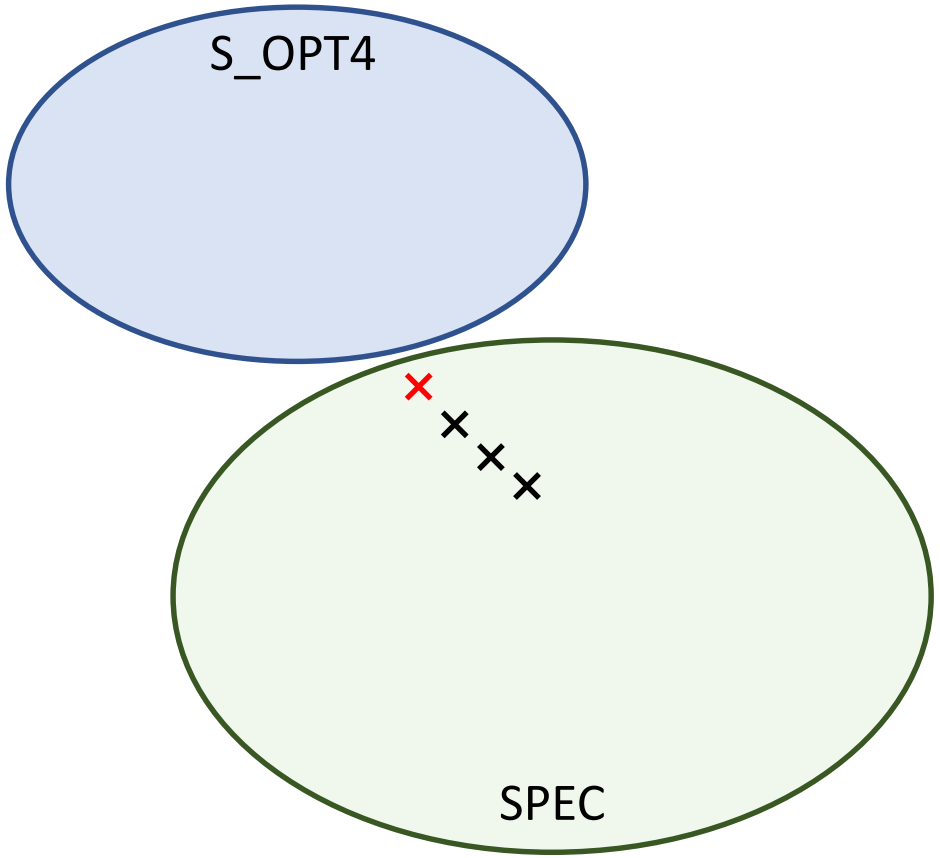
Why?
Because better solutions are unrealizable

Hu et al. Syntax-Guided Synthesis with Quantitative Syntactic Objectives, CAV '18

Motivations for Unrealizability: Optimization

Search Space
S
S_OPT1
S_OPT2
S_OPT3
S_OPT4

Behavioral Spec
SPEC



Guaranteed to be optimal solution

Why?
Because better solutions are unrealizable

To prove optimality:
Must be capable of proving unrealizability!

Hu et al. Syntax-Guided Synthesis with Quantitative Syntactic Objectives, CAV '18

Motivations for Unrealizability: Symbolic Exec

```
x = 42;
y = x + 42;
while (x > y)
{
  …
}
assert(f(x,y))
```


Mechtaev et al. Symbolic Execution with Existential Second-Order Constraints, ESEC/FSE '18

Motivations for Unrealizability: Symbolic Exec

```
x = 42;
y = x + 42;
while (x > y)
{
  …
}
assert(f(x,y))
```


Mechtaev et al. Symbolic Execution with Existential Second-Order Constraints, ESEC/FSE '18

Motivations for Unrealizability: Symbolic Exec

```
x = 42;
y = [          ]
while (x > y)
{
  …
}
assert(f(x,y))
```


Mechtaev et al. Symbolic Execution with Existential Second-Order Constraints, ESEC/FSE '18

Motivations for Unrealizability: Symbolic Exec

```
x = 42;
y = [          ]
while (x > y)
{
  …
}
assert(f(x,y))
```

Search Space
LIA

Mechtaev et al. Symbolic Execution with Existential Second-Order Constraints, ESEC/FSE '18

Motivations for Unrealizability: Symbolic Exec

Search Space
LIA

```
x = 42;
y = [          ]
while (x > y)
{
  …
}
assert(f(x,y))
```

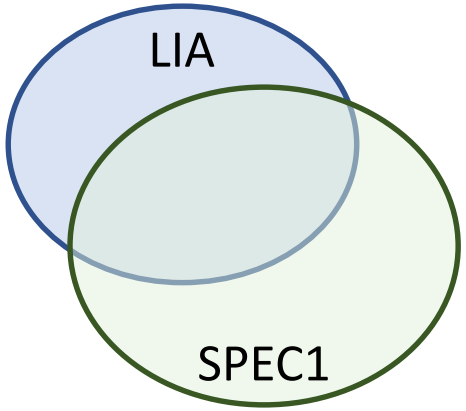
Path Constraints +
Assertion = Spec

Mechtaev et al. Symbolic Execution with Existential Second-Order Constraints, ESEC/FSE '18

Motivations for Unrealizability: Symbolic Exec

```
x = 42;
y = [          ]
while (x > y)
{
  …
}
assert(f(x,y))
```

Search Space
LIA



Path Constraints +
Assertion = Spec

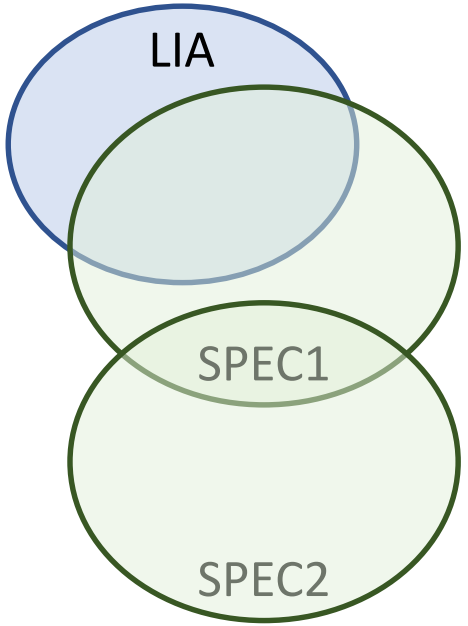
Mechtaev et al. Symbolic Execution with Existential Second-Order Constraints, ESEC/FSE '18

Motivations for Unrealizability: Symbolic Exec

```
x = 42;
y = [          ]
while (x > y)
{
  …
}
assert(f(x,y))
```

Search Space
LIA

Path Constraints +
Assertion = Spec



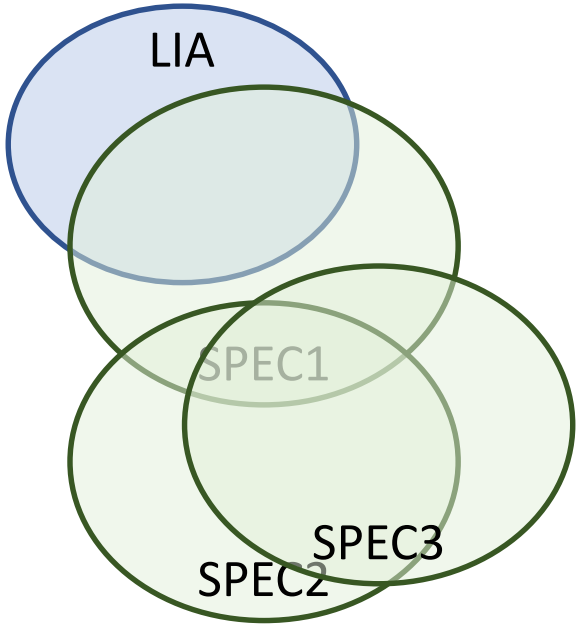
Mechtaev et al. Symbolic Execution with Existential Second-Order Constraints, ESEC/FSE '18

Motivations for Unrealizability: Symbolic Exec

```
x = 42;
y = [          ]
while (x > y)
{
  …
}
assert(f(x,y))
```

Search Space
LIA

Path Constraints +
Assertion = Spec



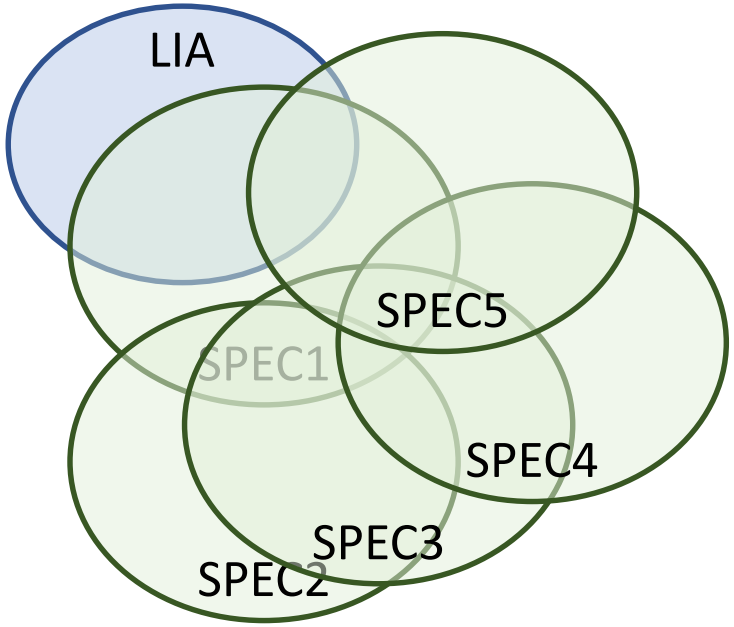
Mechtaev et al. Symbolic Execution with Existential Second-Order Constraints, ESEC/FSE '18

Motivations for Unrealizability: Symbolic Exec

```
x = 42;
y = [          ]
while (x > y)
{
  …
}
assert(f(x,y))
```

Search Space
LIA

Path Constraints +
Assertion = Spec

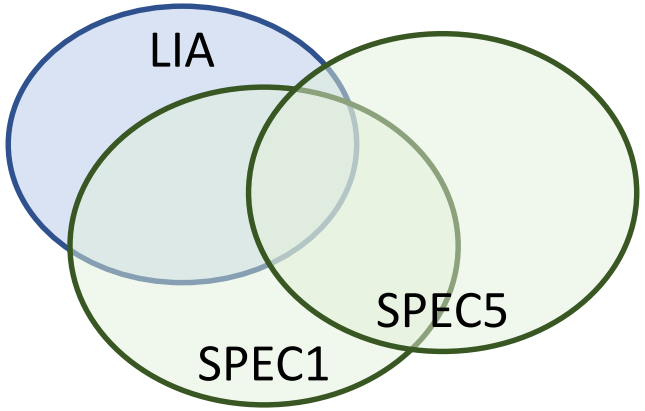


Mechtaev et al. Symbolic Execution with Existential Second-Order Constraints, ESEC/FSE '18

Motivations for Unrealizability: Symbolic Exec

```
x = 42;
y = [          ]
while (x > y)
{
  …
}
assert(f(x,y))
```

Search Space
LIA



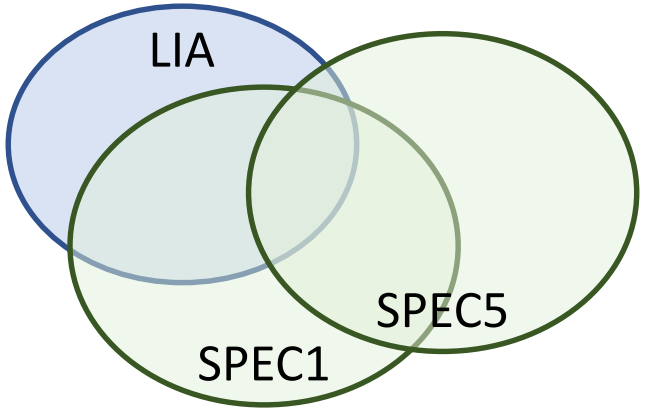
Path Constraints +
Assertion = Spec

Mechtaev et al. Symbolic Execution with Existential Second-Order Constraints, ESEC/FSE '18

Motivations for Unrealizability: Symbolic Exec

```
x = 42;
y = [          ]
while (x > y)
{
  …
}
assert(f(x,y))
```

Search Space
LIA



Path Constraints +
Assertion = Spec

Unrealizability allows us to prune infeasible program paths in SymEx

Mechtaev et al. Symbolic Execution with Existential Second-Order Constraints, ESEC/FSE '18

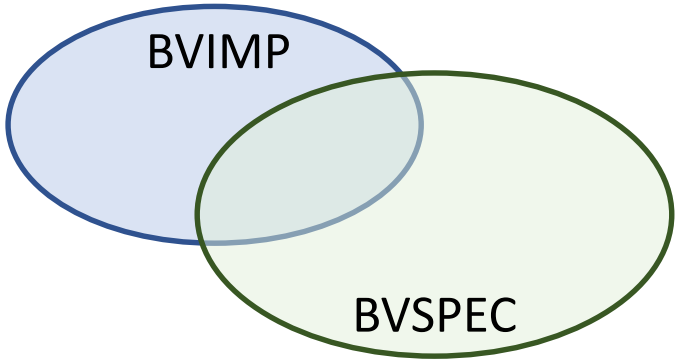
Motivations for Unrealizability: Synthesis

BVIMP

$S \to x := BV \mid y := BV \mid S; S$

$BV \to x \mid y \mid BV \,\&\, BV \mid$

$(BV \mid BV) \mid !\, BV$



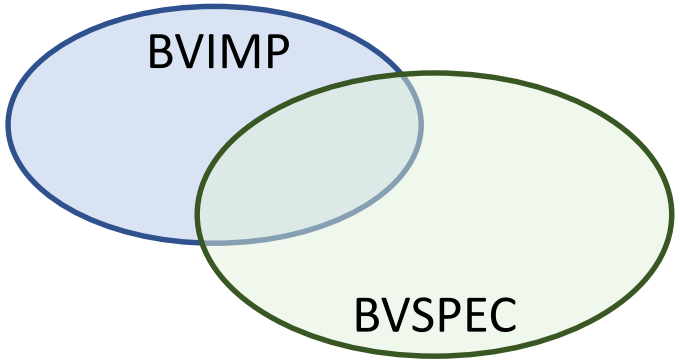
Motivations for Unrealizability: Synthesis

BVIMP

$S \to x := BV \mid y := BV \mid S; S$

$BV \to x \mid y \mid BV \,\&\, BV \mid$

$(BV \mid BV) \mid !\, BV$

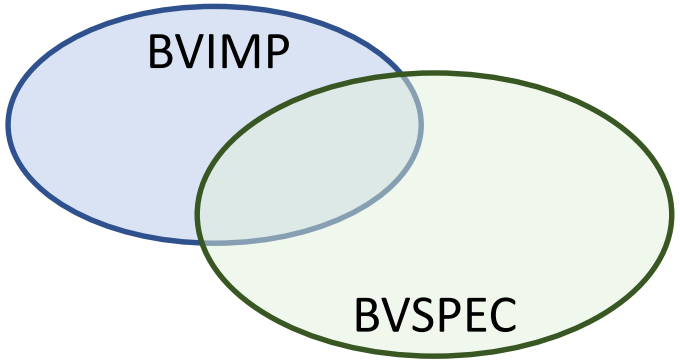


Motivations for Unrealizability: Synthesis

BVIMP

$S \to x := BV \mid y := BV \mid S; S$

$BV \to x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid !\, BV$



$y := BV$

BV

Motivations for Unrealizability: Synthesis

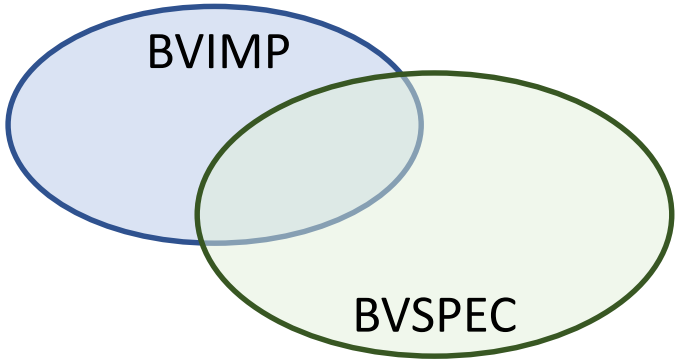
BVIMP

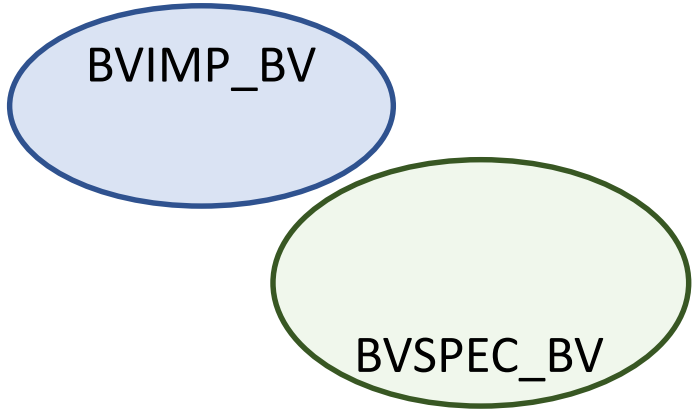
$S \rightarrow x := BV \mid y := BV \mid S; S$

$BV \rightarrow x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid !\, BV$

$y := BV$

BV





Motivations for Unrealizability: Synthesis

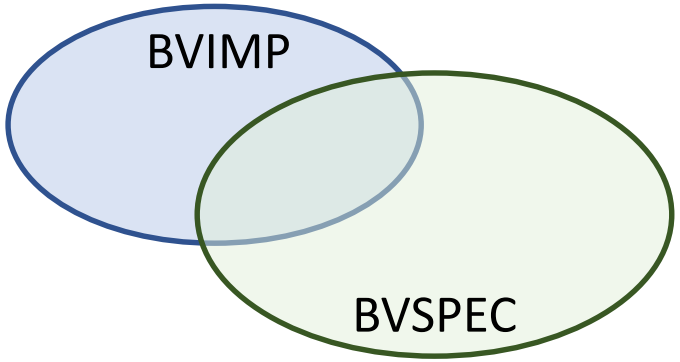
BVIMP

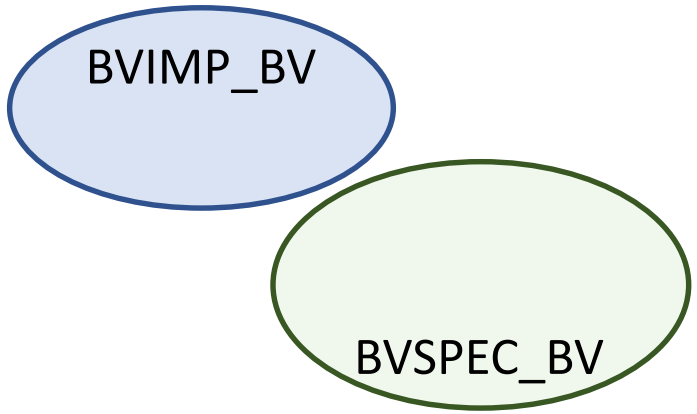
$S \to x := BV \mid \boxed{y := BV} \mid S; S$

$BV \to x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid !\, BV$

$y := BV$

$\boxed{BV}$





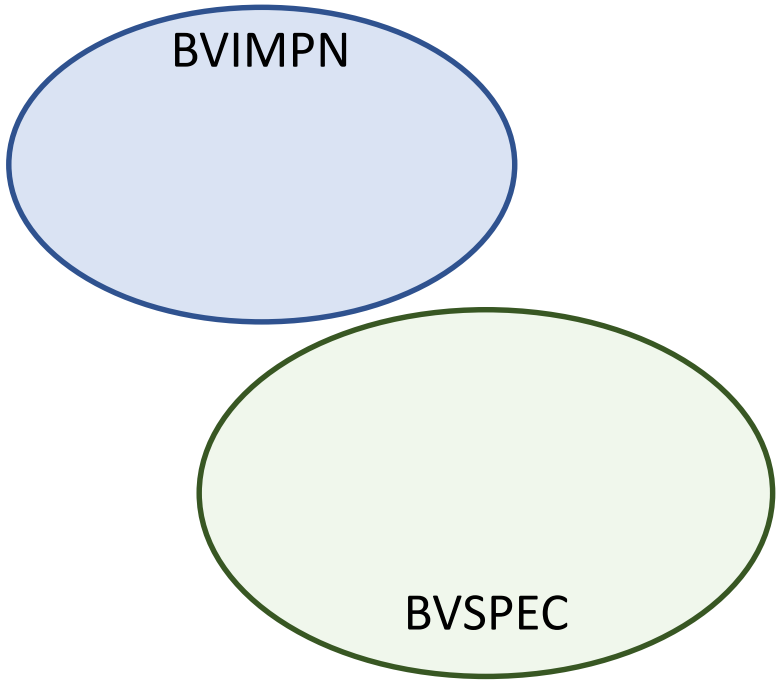
Unrealizability allows us to prune spurious parts of the search space!

Sometimes, there is no such f: Unrealizability

Search Space

$S \to x := BV \mid y := BV \mid S; S$
$BV \to x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid$ ~~$!\, BV$~~

BVIMPN



Behavioral Spec

$f(x, y) = (x \oplus y, y)$

$f(1, 10) = (11, 10) \wedge$
$f(14,15) = (1,15)$

BVSPEC

Goal: Find f in the intersection

~~$x := (x \,\&\, !\, y) \mid (!\, x \,\&\, y)$~~

Now: No such f in the intersection! (Unrealizability)

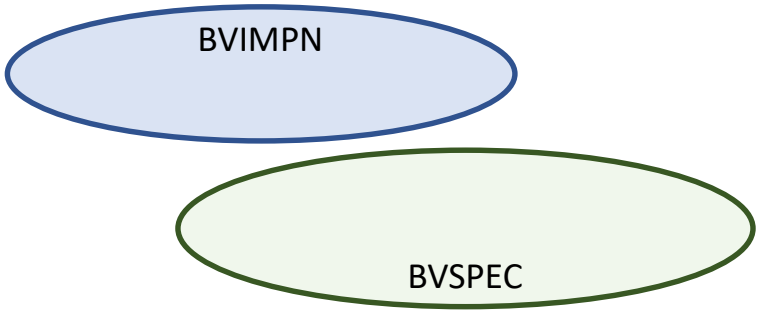
Unrealizability as a Logical Formula

Search Space

$S \to x := BV \mid y := BV \mid S; S$
$BV \to x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid \sout{!BV}$ BVIMPN

Behavioral Spec

$f(x, y) = (x \oplus y, y)$
$f(1, 10) = (11, 10) \wedge$
$f(14{,}15) = (1{,}15)$ BVSPEC



Goal: Find f in the intersection

$\sout{x := (x \,\&\, !\,y) \mid (!\,x \,\&\, y)}$

Now: No such f in the intersection! (Unrealizability)

$$\forall f.\, f \in L(S), \exists x.\, \neg\phi(f(x), x)$$

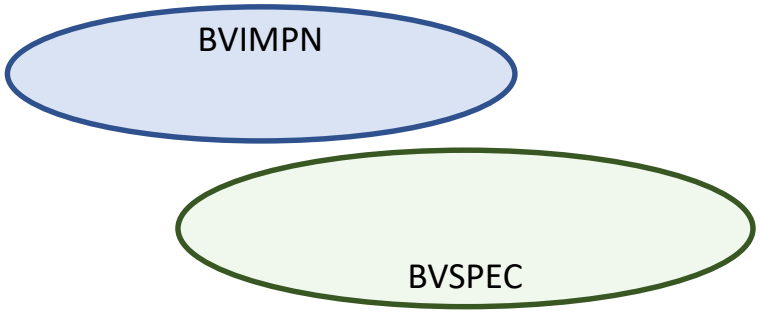
The Challenges of Proving Unrealizability

Search Space

$S \to x := BV \mid y := BV \mid S; S$
$BV \to x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid !\,\cancel{BV}$ BVIMPN

Behavioral Spec

$f(x, y) = (x \oplus y, y)$
$f(1, 10) = (11, 10) \wedge$
$f(14{,}15) = (1{,}15)$ BVSPEC



Goal: Find f in the intersection

$\cancel{x := (x \,\&\, !\,y) \mid (!\,x \,\&\, y)}$

Now: No such f in the intersection! (Unrealizability)

Must reason about the behavior of a (possibly infinite) *set* of programs…

$$\forall f.\, f \in L(S), \exists x.\, \neg\phi(f(x), x)$$

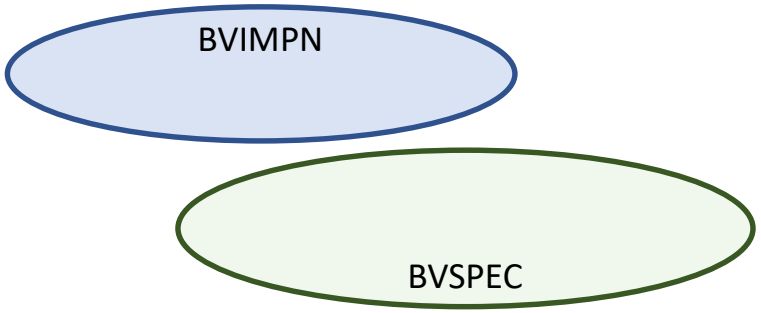
The Challenges of Proving Unrealizability

Search Space

$S \to x := BV \mid y := BV \mid S; S$
$BV \to x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid \sout{!\,BV}$ BVIMPN

Behavioral Spec

$f(x, y) = (x \oplus y, y)$
$f(1, 10) = (11, 10) \wedge$
$f(14{,}15) = (1{,}15)$ BVSPEC



Goal: Find f in the intersection

$\sout{x := (x \,\&\, !\,y) \mid (!\,x \,\&\, y)}$

Now: No such f in the intersection! (Unrealizability)

Must reason about the behavior of a (possibly infinite) *set* of programs…

To guarantee that there exists some inputs (from a possibly infinite set)…

$$\forall f.\, f \in L(S), \exists x.\, \neg\phi(f(x), x)$$

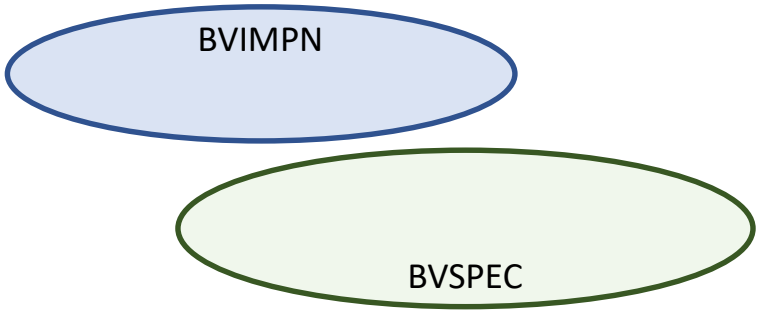
The Challenges of Proving Unrealizability

Search Space

$S \to x := BV \mid y := BV \mid S; S$
$BV \to x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid \sout{!BV}$ BVIMPN

Behavioral Spec

$f(x, y) = (x \oplus y, y)$
$f(1, 10) = (11, 10) \wedge$
$f(14{,}15) = (1{,}15)$ BVSPEC



Goal: Find f in the intersection

$\sout{x := (x \,\&\, !\,y) \mid (!\,x \,\&\, y)}$

Now: No such f in the intersection! (Unrealizability)

Must reason about the behavior of a (possibly infinite) *set* of programs…

To guarantee that there exists some inputs (from a possibly infinite set)…

$$\forall f.\, f \in L(S), \exists x.\, \neg\phi(f(x), x)$$

That falsifies the desired spec…

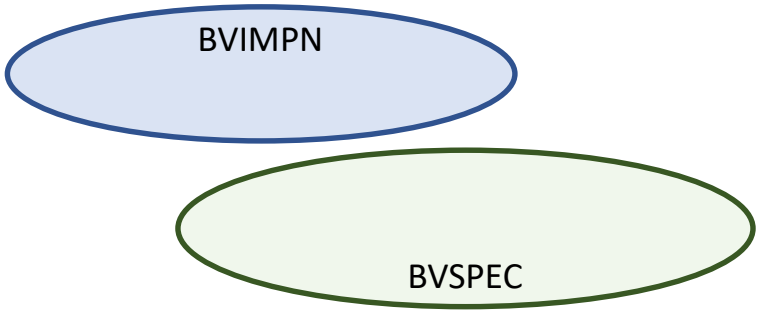
The Challenges of Proving Unrealizability

Search Space

$S \to x := BV \mid y := BV \mid S; S$
$BV \to x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid \sout{!BV}$ BVIMPN

Behavioral Spec

$f(x, y) = (x \oplus y, y)$
$f(1, 10) = (11, 10) \wedge$
$f(14{,}15) = (1{,}15)$ BVSPEC



Goal: Find f in the intersection

$\sout{x := (x \,\&\, !\,y) \mid (!\,x \,\&\, y)}$

Now: No such f in the intersection! (Unrealizability)

Must reason about the behavior of a (possibly infinite) *set* of programs…

To guarantee that there exists some inputs (from a possibly infinite set)…

$$\forall f.\, f \in L(S), \exists x.\, \neg\phi(f(x), x)$$

And the falsifying input may be different for each program!

That falsifies the desired spec…

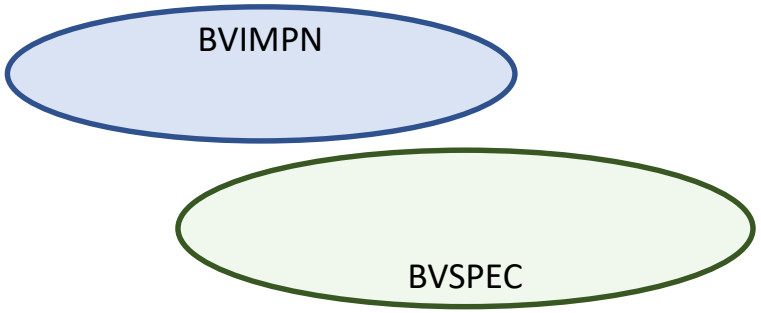
The Challenges of Proving Unrealizability

Search Space

$S \to x := BV \mid y := BV \mid S; S$
$BV \to x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid \sout{!BV}$ BVIMPN

Behavioral Spec

$f(x, y) = (x \oplus y, y)$
$f(1, 10) = (11, 10) \wedge$
$f(14{,}15) = (1{,}15)$ BVSPEC



Goal: Find f in the intersection

$\sout{x := (x \,\&\, !\,y) \mid (!\,x \,\&\, y)}$

Now: No such f in the intersection! (Unrealizability)

Must reason about the behavior of a (possibly infinite) *set* of programs…

To guarantee that there exists some inputs (from a possibly infinite set)…

$$\forall f.\, f \in L(S), \exists x.\, \neg\phi(f(x), x)$$

And the falsifying input may be different for each program!

That falsifies the desired spec…

Aspect 1: Reasoning about Sets of Programs

Dealing with Sets of Programs: Unrealizability Triples

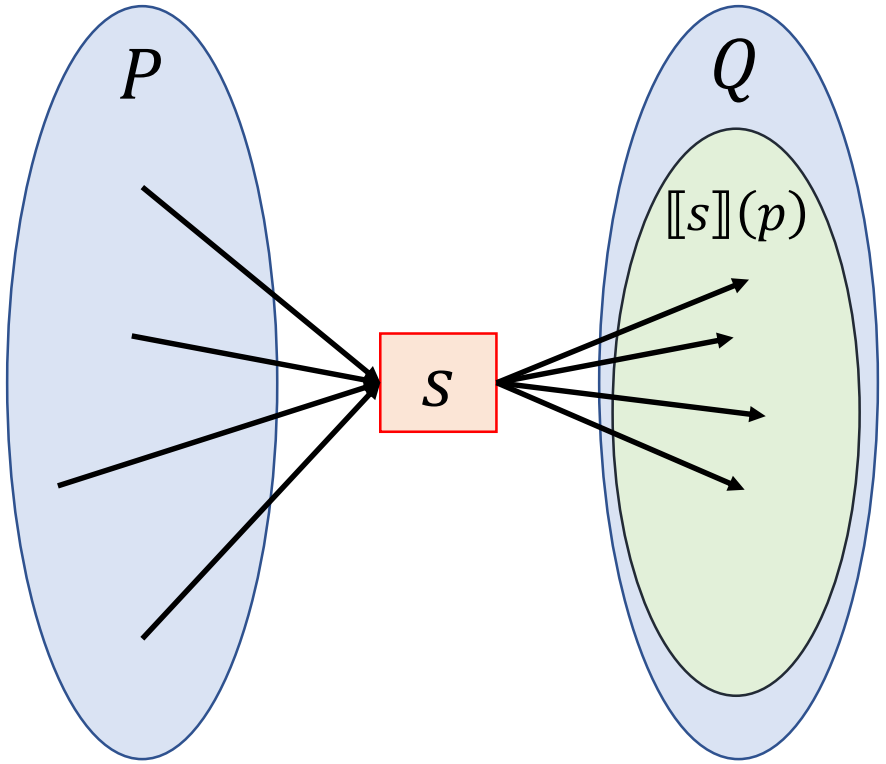
$$\{P\}\ s\ \{Q\}$$

$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$

Dealing with Sets of Programs: Unrealizability Triples

$$\{P\}\ s\ \{Q\}$$

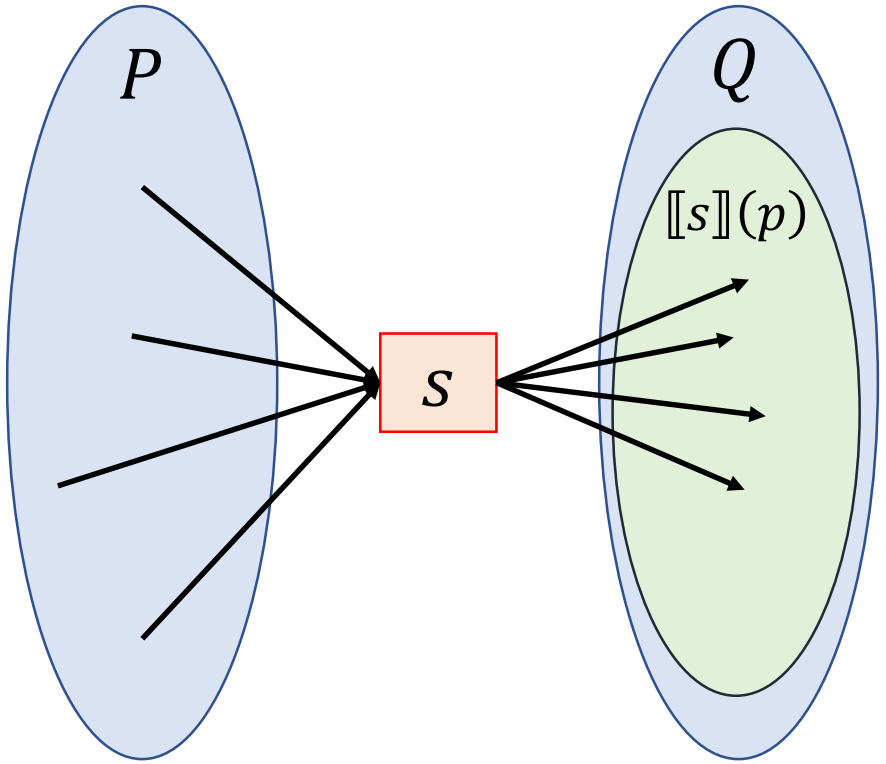
$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$



Dealing with Sets of Programs: Unrealizability Triples

$$\{P\}\ s\ \{Q\}$$

$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$

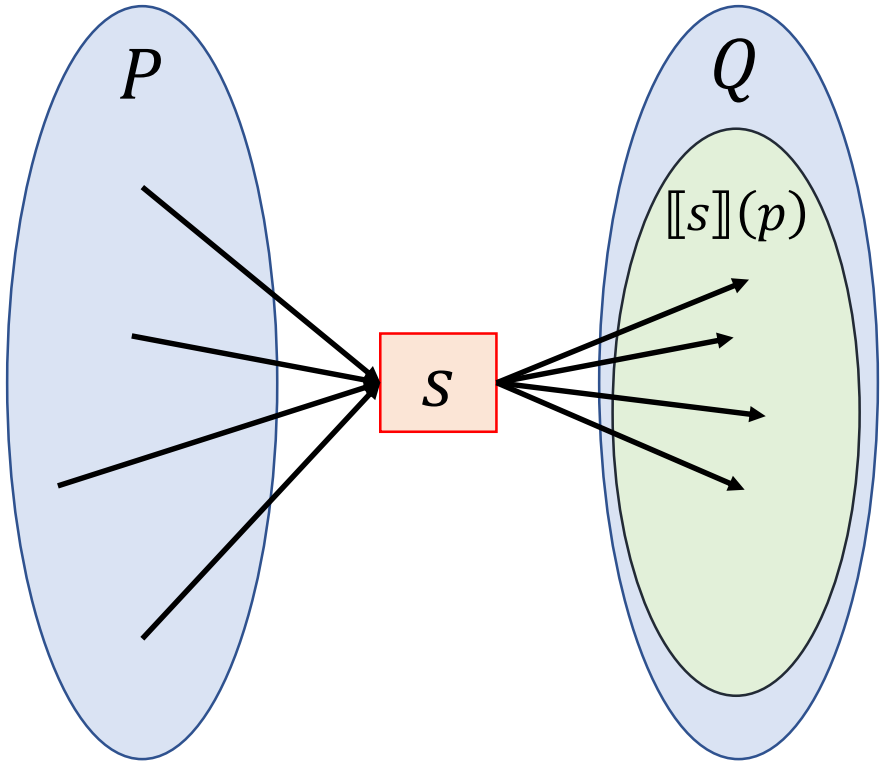


$$\{|P|\}\ S\ \{|Q|\}$$

Dealing with Sets of Programs: Unrealizability Triples

$$\{P\}\ s\ \{Q\}$$

$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$



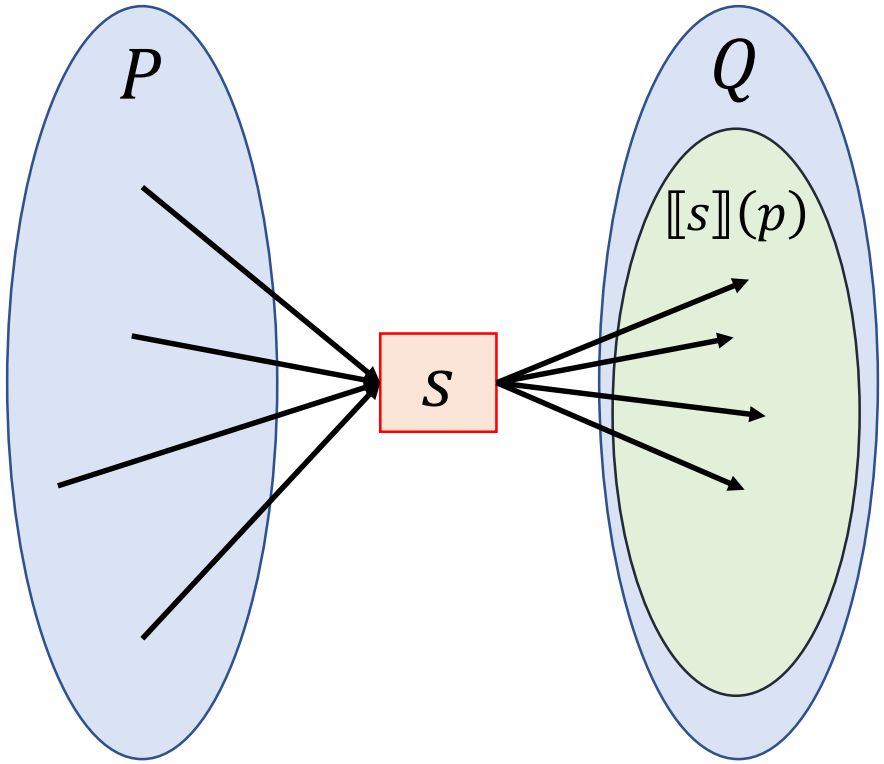
$$\{|P|\}\ S\ \{|Q|\}$$

$$\forall p \in P, s \in S.\, \exists q \in Q.\, [\![s]\!](p) = q$$

Dealing with Sets of Programs: Unrealizability Triples

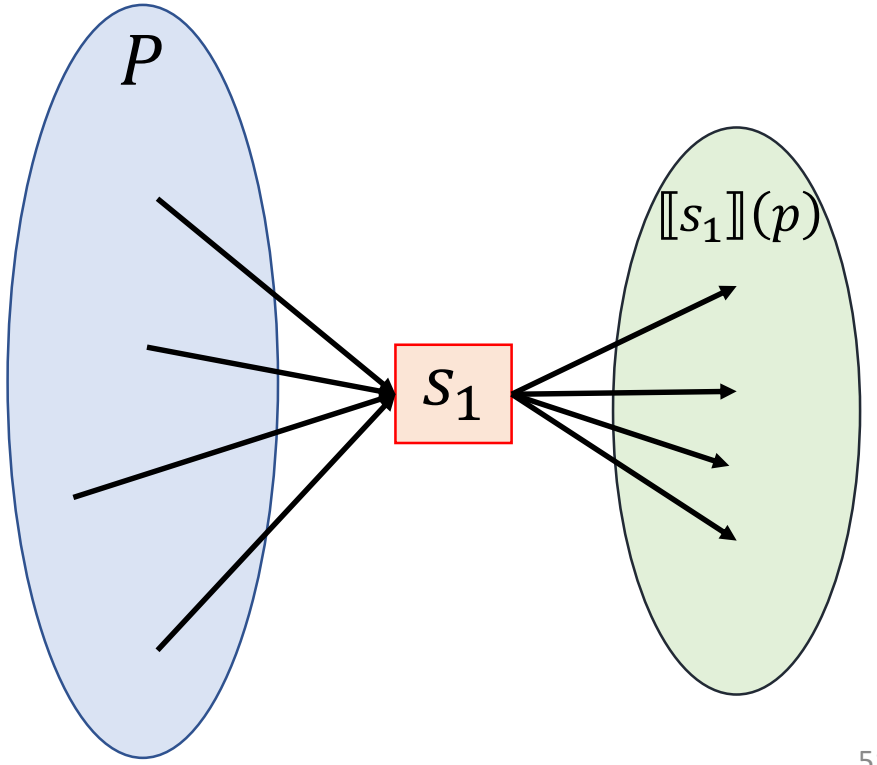
$$\{P\}\ s\ \{Q\}$$

$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$



$$\{|P|\}\ S\ \{|Q|\}$$

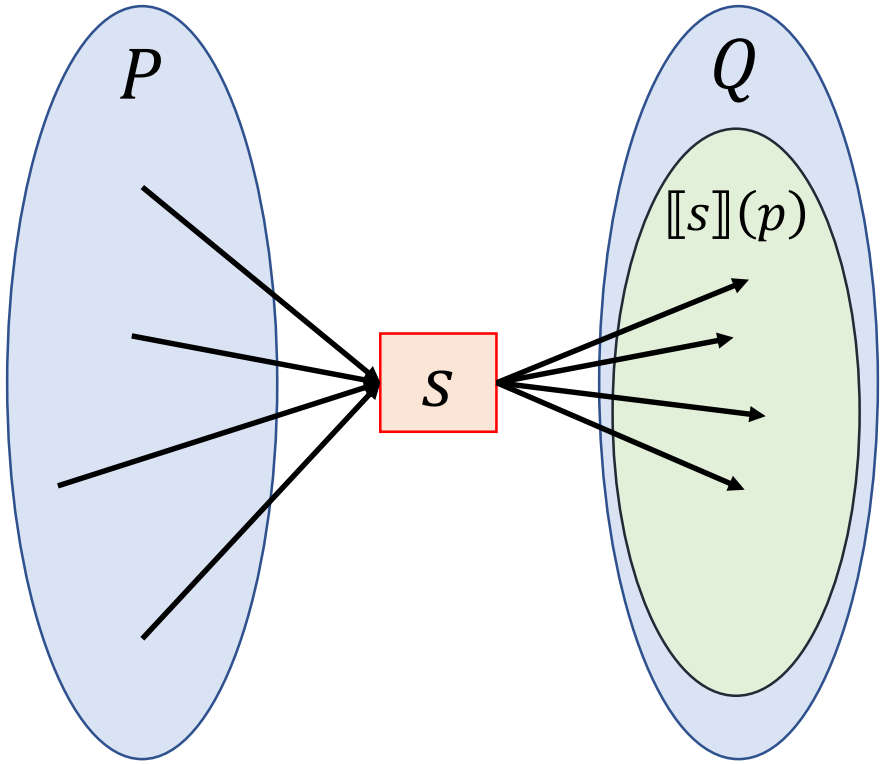
$$\forall p \in P, s \in S.\, \exists q \in Q.\, [\![s]\!](p) = q$$



Dealing with Sets of Programs: Unrealizability Triples

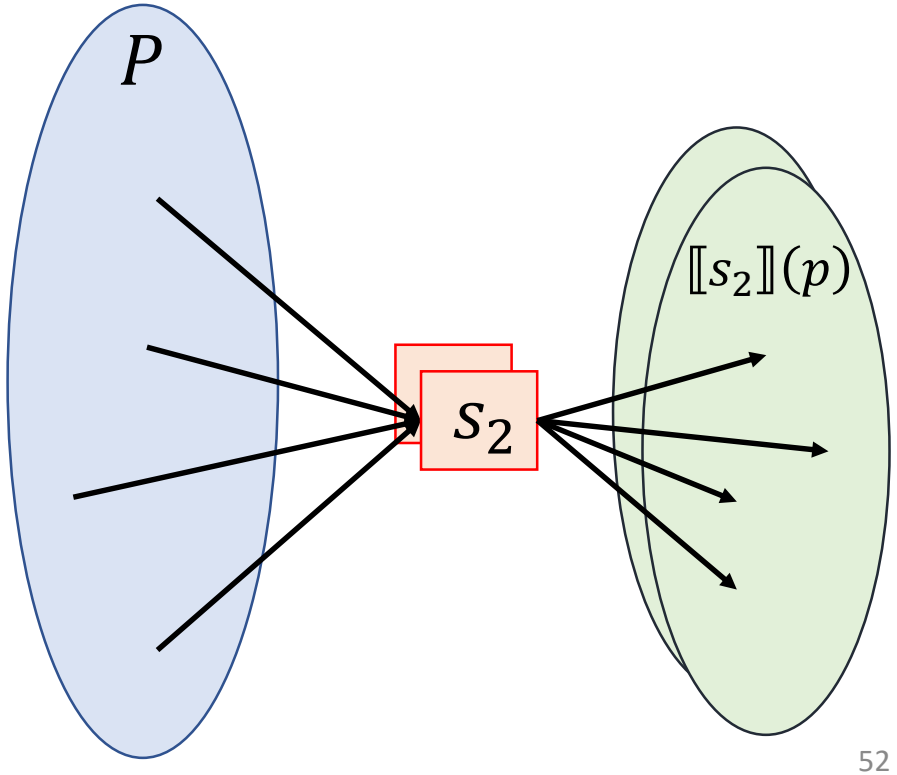
$$\{P\}\ s\ \{Q\}$$

$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$



$$\{|P|\}\ S\ \{|Q|\}$$

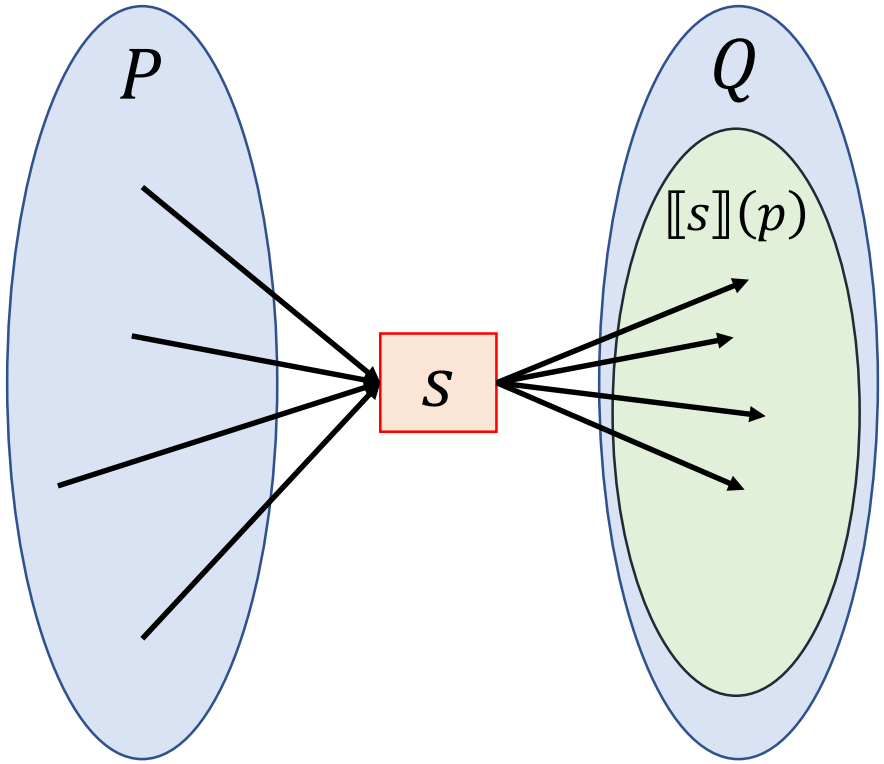
$$\forall p \in P, s \in S.\, \exists q \in Q.\, [\![s]\!](p) = q$$



Dealing with Sets of Programs: Unrealizability Triples

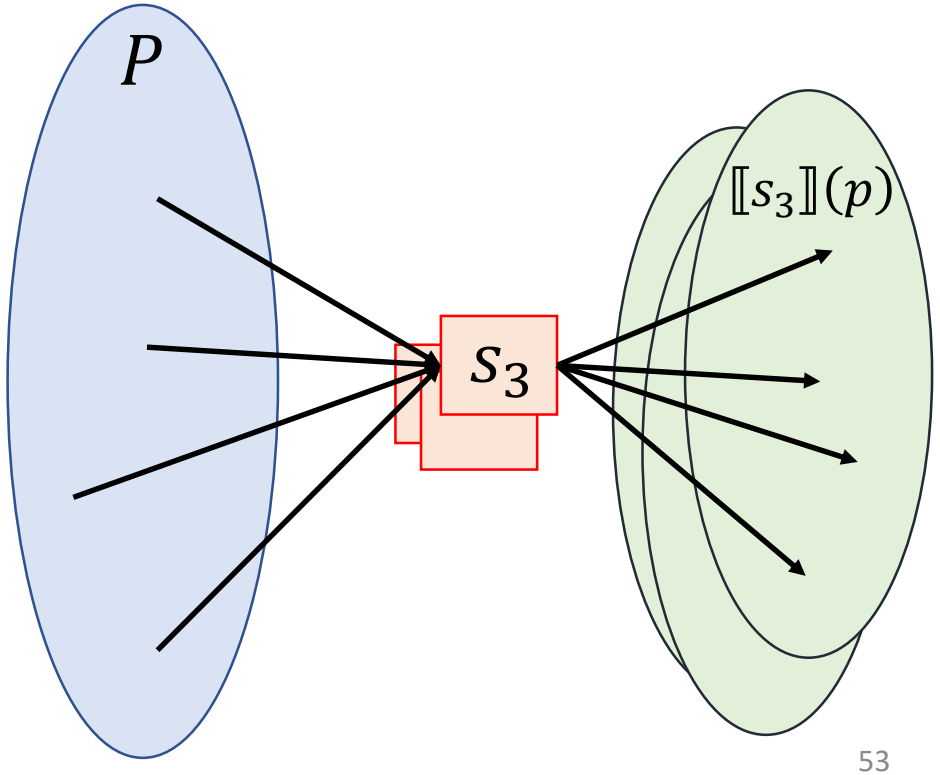
$$\{P\}\ s\ \{Q\}$$

$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$



$$\{|P|\}\ S\ \{|Q|\}$$

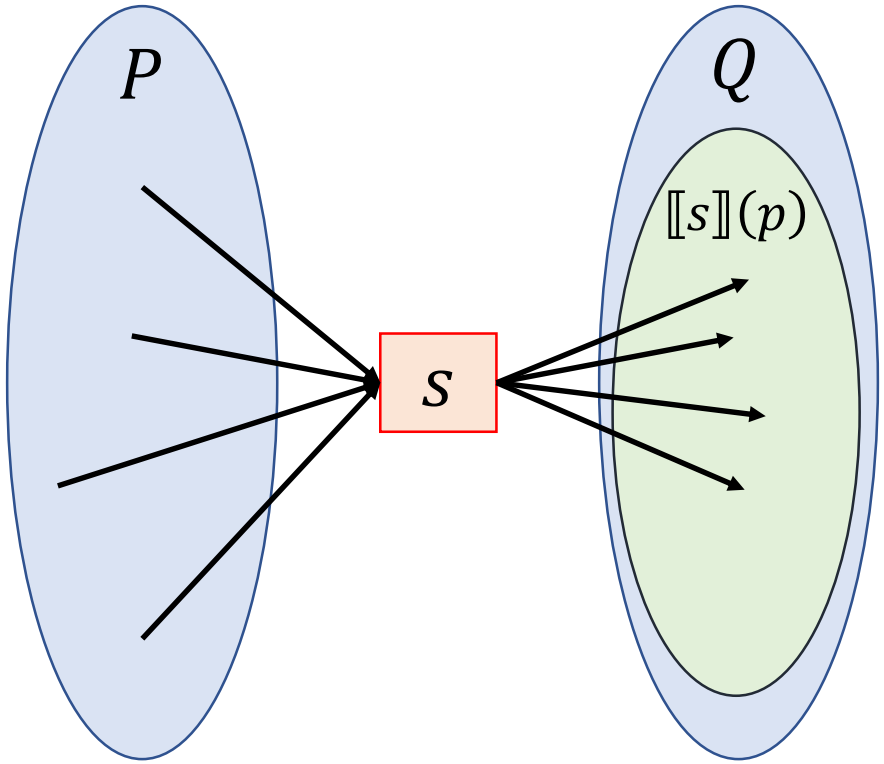
$$\forall p \in P, s \in S.\, \exists q \in Q.\, [\![s]\!](p) = q$$



Dealing with Sets of Programs: Unrealizability Triples

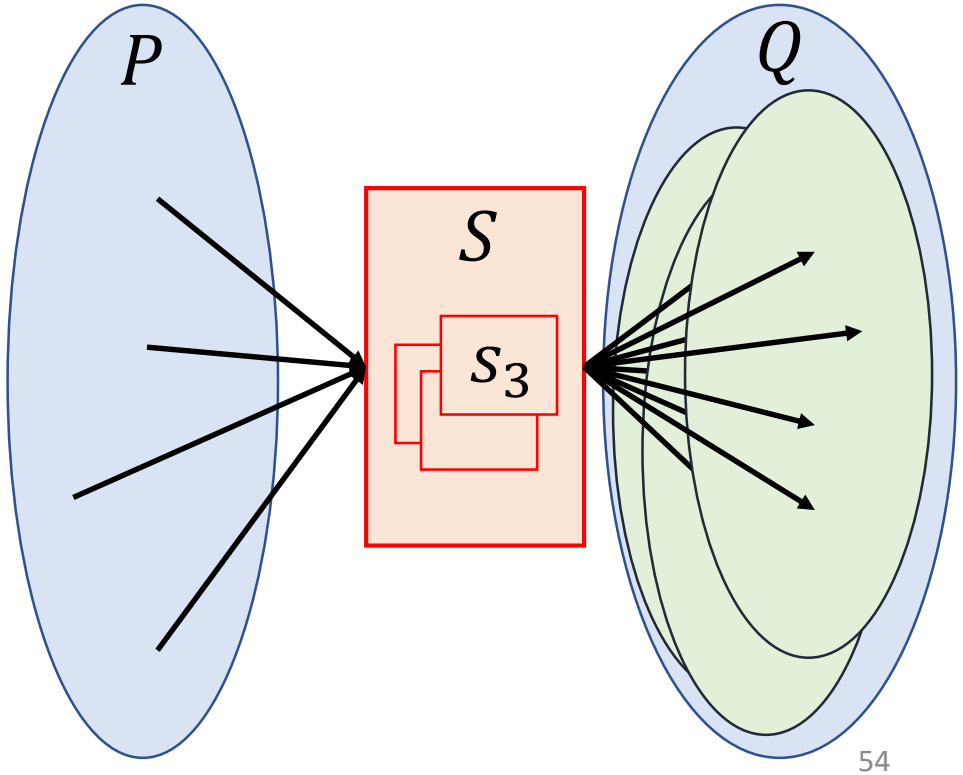
$$\{P\}\ s\ \{Q\}$$

$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$



$$\{|P|\}\ S\ \{|Q|\}$$

$$\forall p \in P, s \in S.\, \exists q \in Q.\, [\![s]\!](p) = q$$



Dealing with Sets of Programs: Unrealizability Triples

$\{P\}\ s\ \{Q\}$

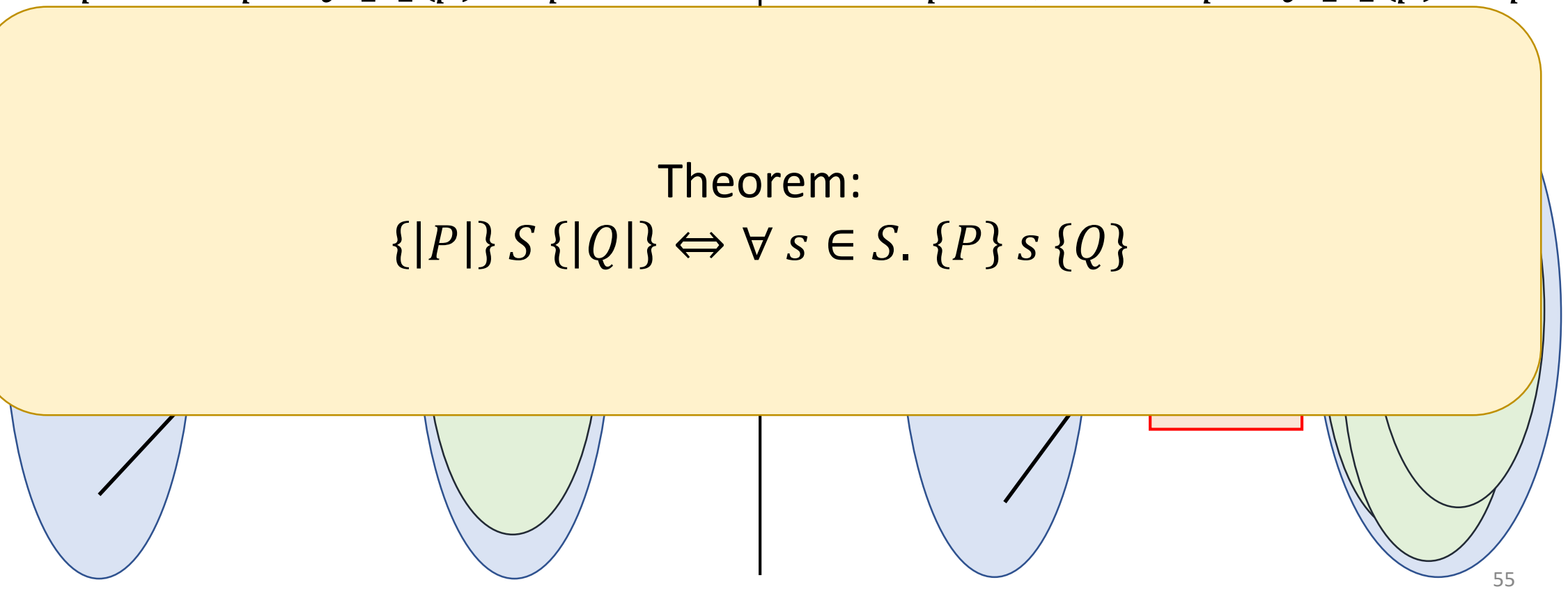
$\forall p \in P. \exists q \in Q. [\![s]\!](p) = q$

$\{|P|\}\ S\ \{|Q|\}$

$\forall p \in P, s \in S. \exists q \in Q. [\![s]\!](p) = q$

Theorem:

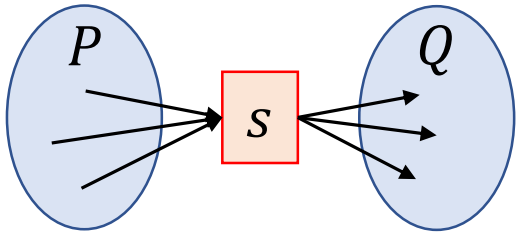
$\{|P|\}\ S\ \{|Q|\} \Leftrightarrow \forall\ s \in S.\ \{P\}\ s\ \{Q\}$



Using Unrealizability Triples: Inference Rules

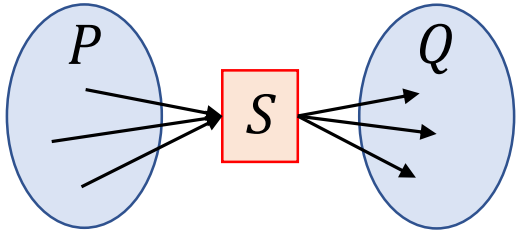
$$\{P\}\ s\ \{Q\}$$

$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$



$$\{|P|\}\ S\ \{|Q|\}$$

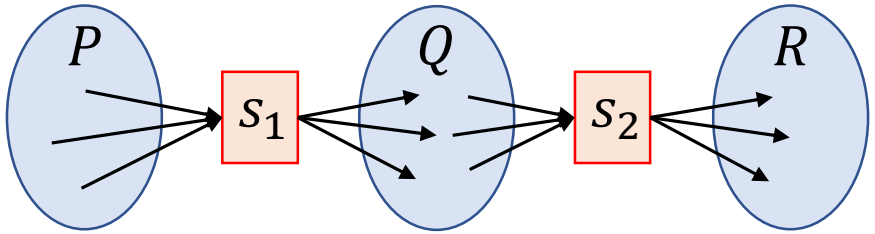
$$\forall p \in P, s \in S.\, \exists q \in Q.\, [\![s]\!](p) = q$$



Using Unrealizability Triples: Inference Rules

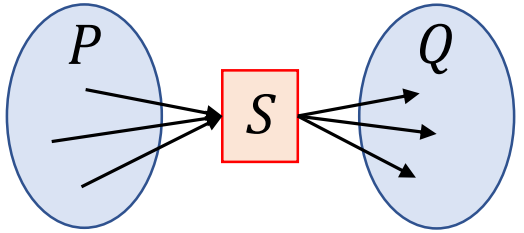
$$\{P\}\ s\ \{Q\}$$

$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$



$$\{|P|\}\ S\ \{|Q|\}$$

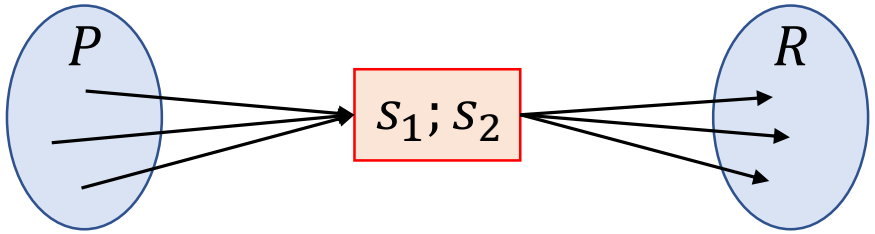
$$\forall p \in P, s \in S.\, \exists q \in Q.\, [\![s]\!](p) = q$$



Using Unrealizability Triples: Inference Rules

$$\{P\}\ s\ \{Q\}$$

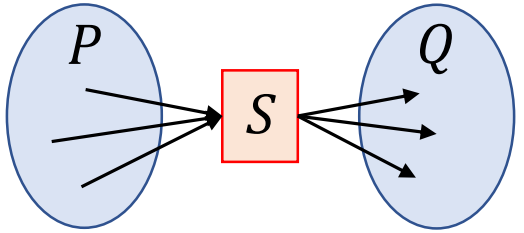
$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$



$$\frac{\{P\}\ s_1\ \{Q\} \qquad \{Q\}\ s_2\ \{R\}}{\{P\}\ s_1; s_2\ \{Q\}}$$

$$\{|P|\}\ S\ \{|Q|\}$$

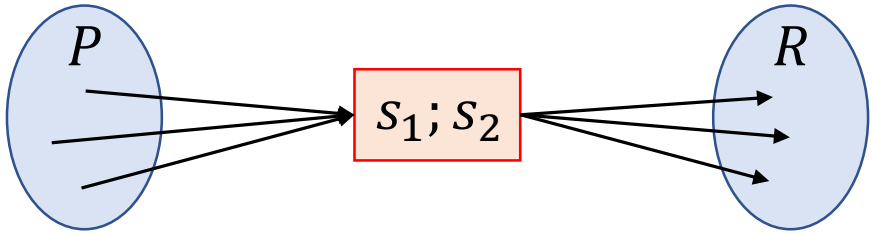
$$\forall p \in P, s \in S.\, \exists q \in Q.\, [\![s]\!](p) = q$$



Using Unrealizability Triples: Inference Rules

$$\{P\}\ s\ \{Q\}$$

$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$

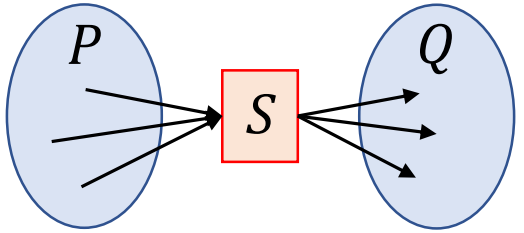


$$\frac{\{P\}\ s_1\ \{Q\} \qquad \{Q\}\ s_2\ \{R\}}{\{P\}\ s_1; s_2\ \{Q\}}$$

Principle of overapproximation allows for compositional reasoning!

$$\{|P|\}\ S\ \{|Q|\}$$

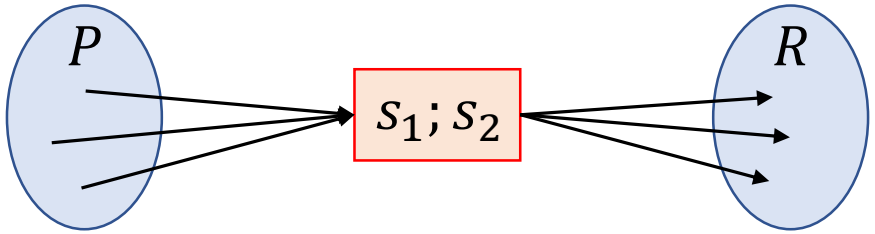
$$\forall p \in P, s \in S.\, \exists q \in Q.\, [\![s]\!](p) = q$$



Using Unrealizability Triples: Inference Rules

$$\{P\}\ s\ \{Q\}$$

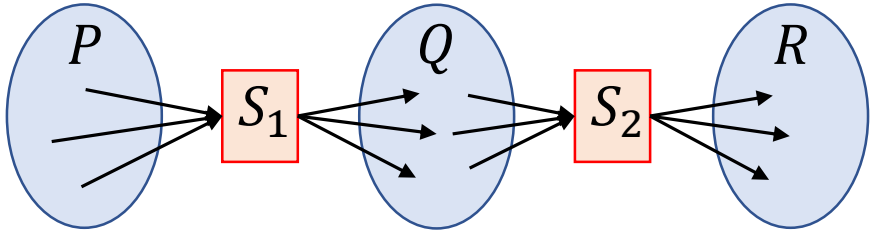
$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$



$$\frac{\{P\}\ s_1\ \{Q\} \qquad \{Q\}\ s_2\ \{R\}}{\{P\}\ s_1; s_2\ \{Q\}}$$

$$\{|P|\}\ S\ \{|Q|\}$$

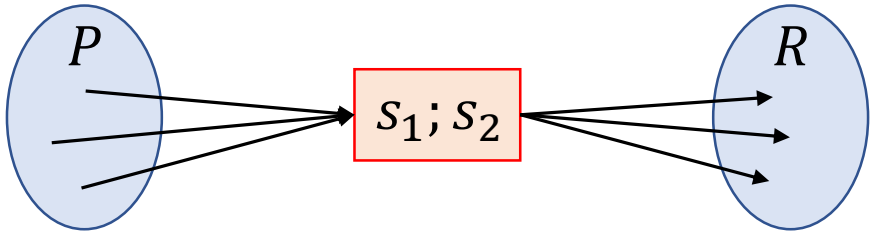
$$\forall p \in P, s \in S.\, \exists q \in Q.\, [\![s]\!](p) = q$$



Using Unrealizability Triples: Inference Rules

$$\{P\}\ s\ \{Q\}$$

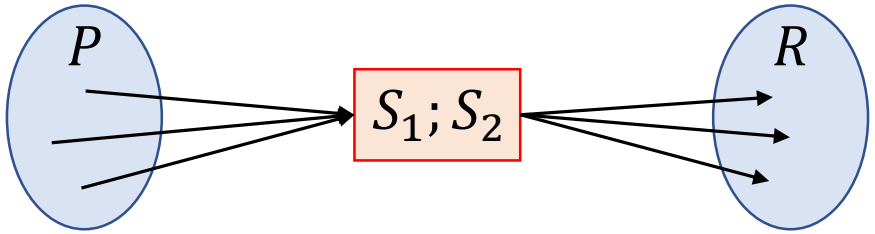
$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$



$$\frac{\{P\}\ s_1\ \{Q\} \qquad \{Q\}\ s_2\ \{R\}}{\{P\}\ s_1; s_2\ \{Q\}}$$

$$\{|P|\}\ S\ \{|Q|\}$$

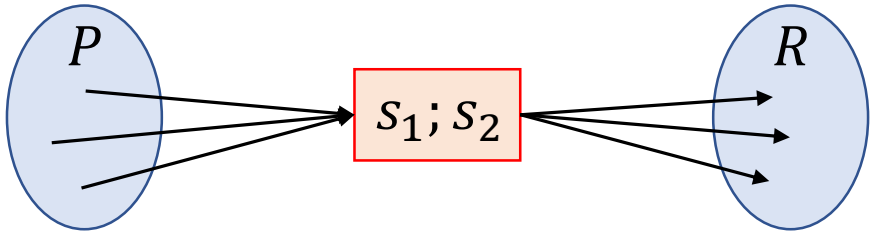
$$\forall p \in P, s \in S.\, \exists q \in Q.\, [\![s]\!](p) = q$$



Using Unrealizability Triples: Inference Rules

$$\{P\}\ s\ \{Q\}$$

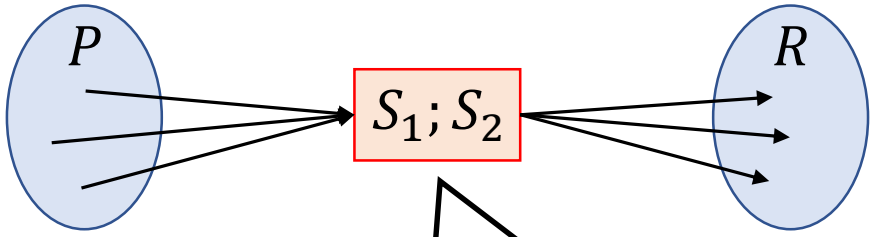
$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$



$$\frac{\{P\}\ s_1\ \{Q\} \qquad \{Q\}\ s_2\ \{R\}}{\{P\}\ s_1; s_2\ \{Q\}}$$

$$\{|P|\}\ S\ \{|Q|\}$$

$$\forall p \in P, s \in S.\, \exists q \in Q.\, [\![s]\!](p) = q$$



Recall BVIMP:

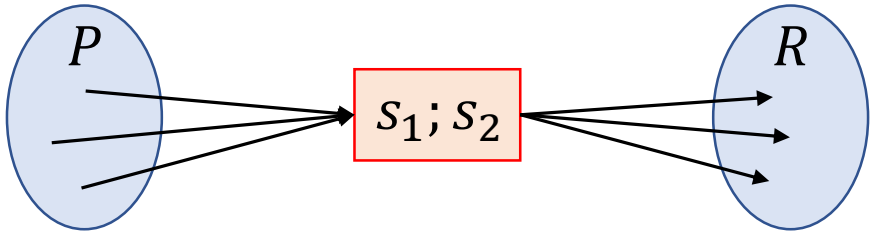
$S \rightarrow x := BV \mid y := BV \mid S; S$

$BV \rightarrow x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid !\, BV$

Using Unrealizability Triples: Inference Rules

$$\{P\}\ s\ \{Q\}$$

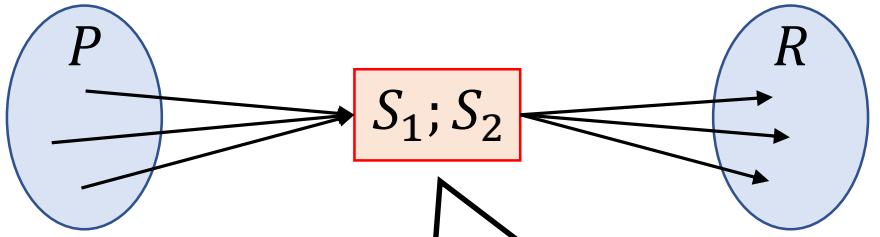
$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$



$$\frac{\{P\}\ s_1\ \{Q\} \qquad \{Q\}\ s_2\ \{R\}}{\{P\}\ s_1; s_2\ \{Q\}}$$

$$\{|P|\}\ S\ \{|Q|\}$$

$$\forall p \in P, s \in S.\, \exists q \in Q.\, [\![s]\!](p) = q$$



Recall BVIMP:

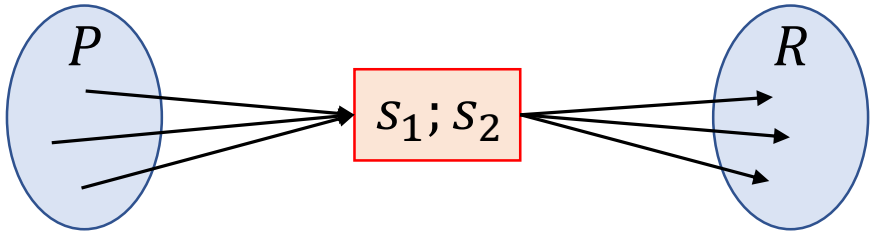
$S \rightarrow x := BV \mid y := BV \mid S; S$

$BV \rightarrow x \mid y \mid BV \,\&\, BV \mid$
$(BV \mid BV) \mid !\, BV$

Using Unrealizability Triples: Inference Rules

$$\{P\}\ s\ \{Q\}$$

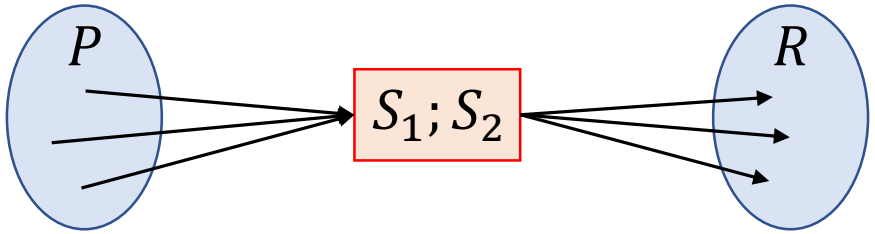
$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$



$$\frac{\{P\}\ s_1\ \{Q\} \qquad \{Q\}\ s_2\ \{R\}}{\{P\}\ s_1; s_2\ \{Q\}}$$

$$\{|P|\}\ S\ \{|Q|\}$$

$$\forall p \in P, s \in S.\, \exists q \in Q.\, [\![s]\!](p) = q$$

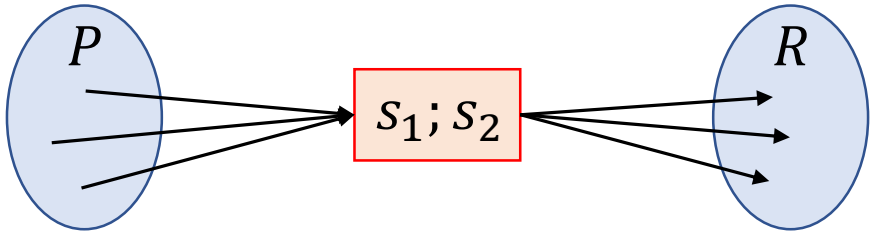


$$\frac{\{|P|\}\ S_1\ \{|Q|\} \qquad \{|Q|\}\ S_2\ \{|R|\}}{\{|P|\}\ S_1; S_2\ \{|R|\}}$$

Using Unrealizability Triples: Inference Rules

$$\{P\}\ s\ \{Q\}$$

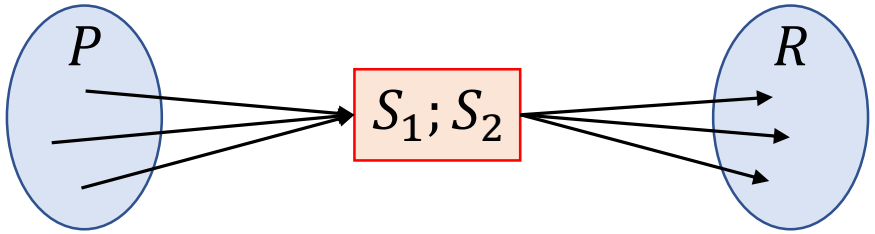
$$\forall p \in P.\, \exists q \in Q.\, [\![s]\!](p) = q$$



$$\frac{\{P\}\ s_1\ \{Q\} \qquad \{Q\}\ s_2\ \{R\}}{\{P\}\ s_1; s_2\ \{Q\}}$$

$$\{|P|\}\ S\ \{|Q|\}$$

$$\forall p \in P, s \in S.\, \exists q \in Q.\, [\![s]\!](p) = q$$



$$\frac{\{|P|\}\ S_1\ \{|Q|\} \qquad \{|Q|\}\ S_2\ \{|R|\}}{\{|P|\}\ S_1; S_2\ \{|R|\}}$$

Overapproximation + Structure of search space (grammar) allows for compositional reasoning!

Aspect 2: Synchronizing Between Inputs

Synchronizing Between Inputs: Vectorized Semantics

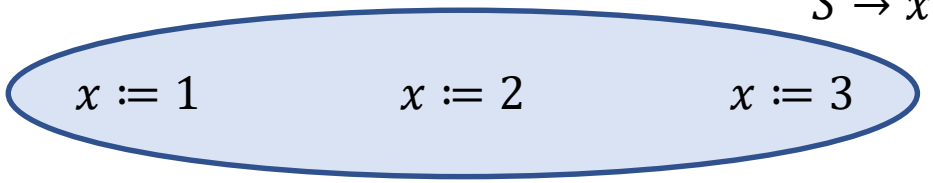
Inputs

$(x = 0, y = 1)$ $(x = 0, y = 2)$ $(x = 0, y = 3)$

Goal: Set the value of x to that in y (e.g., $x := y$)

Language

$S \to x := E$ $E \to 1 \mid E + E$



Synchronizing Between Inputs: Vectorized Semantics

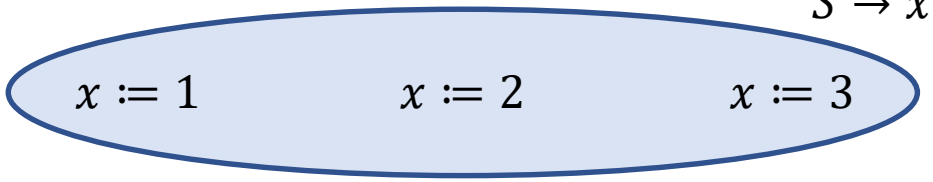
Inputs

$(x = 0, y = 1)$ $(x = 0, y = 2)$ $(x = 0, y = 3)$

Goal: Set the value of x to that in y (e.g., $x := y$)

Language

$S \rightarrow x := E$ $E \rightarrow 1 \mid E + E$



Desired Outputs

$(x = 1, y = 1)$ $(x = 2, y = 2)$ $(x = 3, y = 3)$

Synchronizing Between Inputs: Vectorized Semantics

Inputs

$(x = 0, y = 1)$ $(x = 0, y = 2)$ $(x = 0, y = 3)$

Goal: Set the value of x to that in y (e.g., $x := y$)

Language

$S \to x := E$ $E \to 1 \mid E + E$

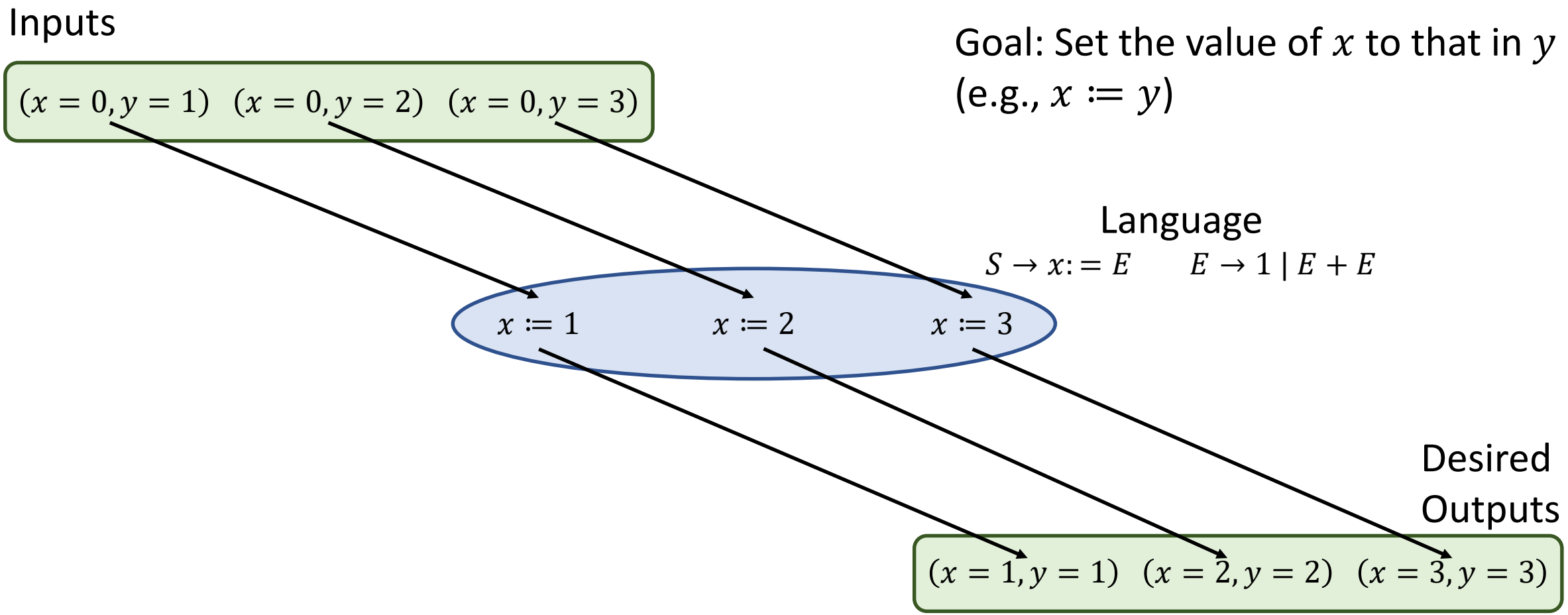
$x := 1$ $x := 2$ $x := 3$

Unrealizable, as only constants may be assigned to x.

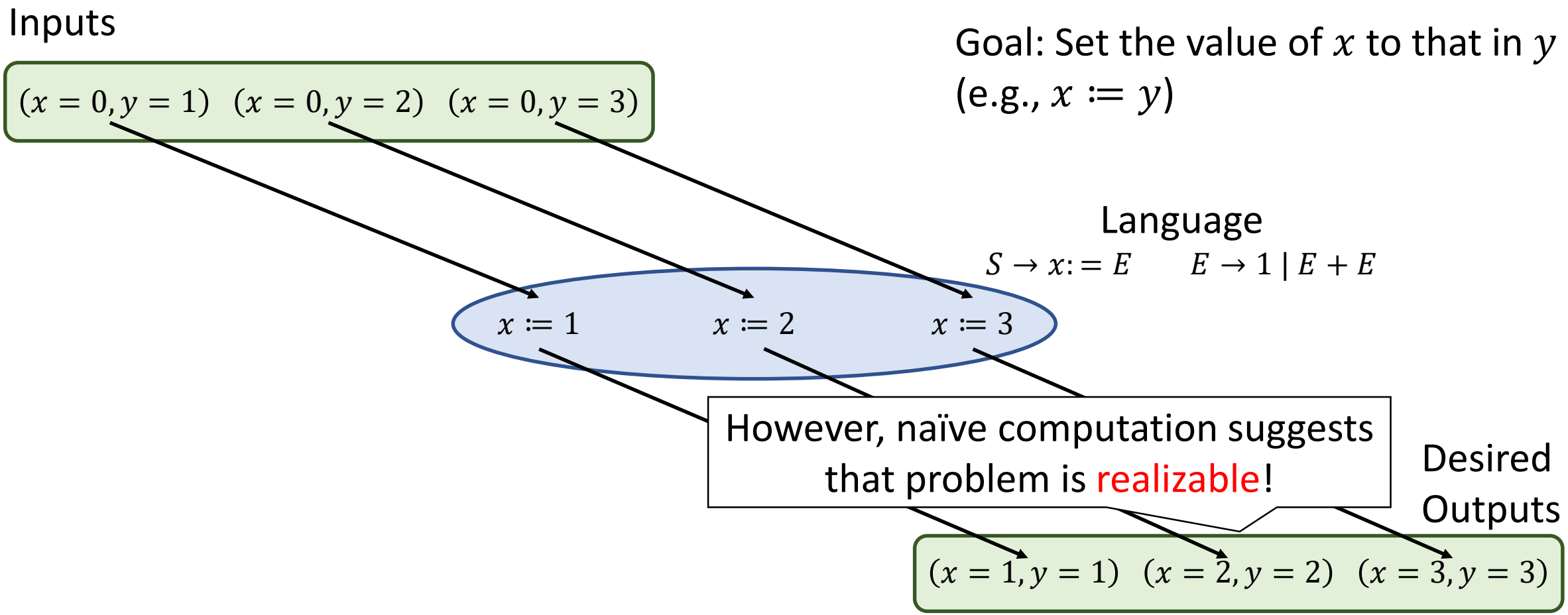
Desired Outputs

$(x = 1, y = 1)$ $(x = 2, y = 2)$ $(x = 3, y = 3)$

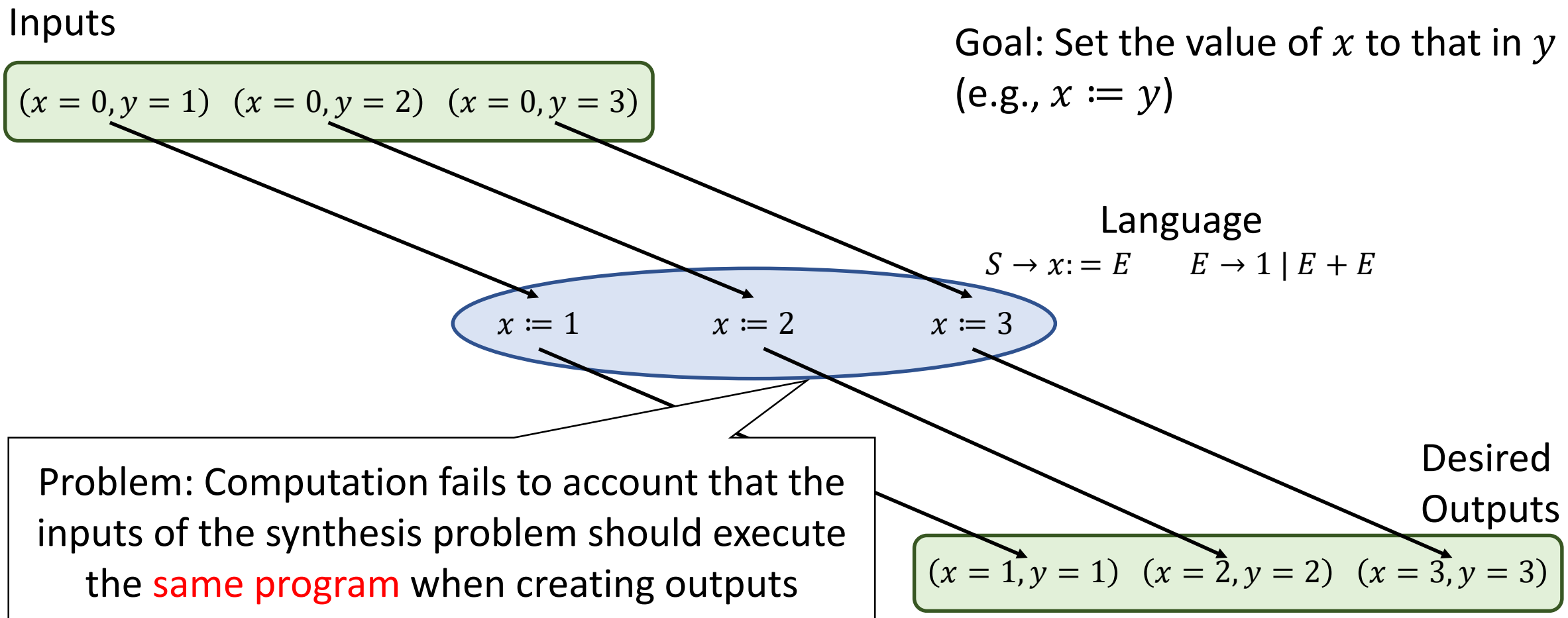
Synchronizing Between Inputs: Vectorized Semantics



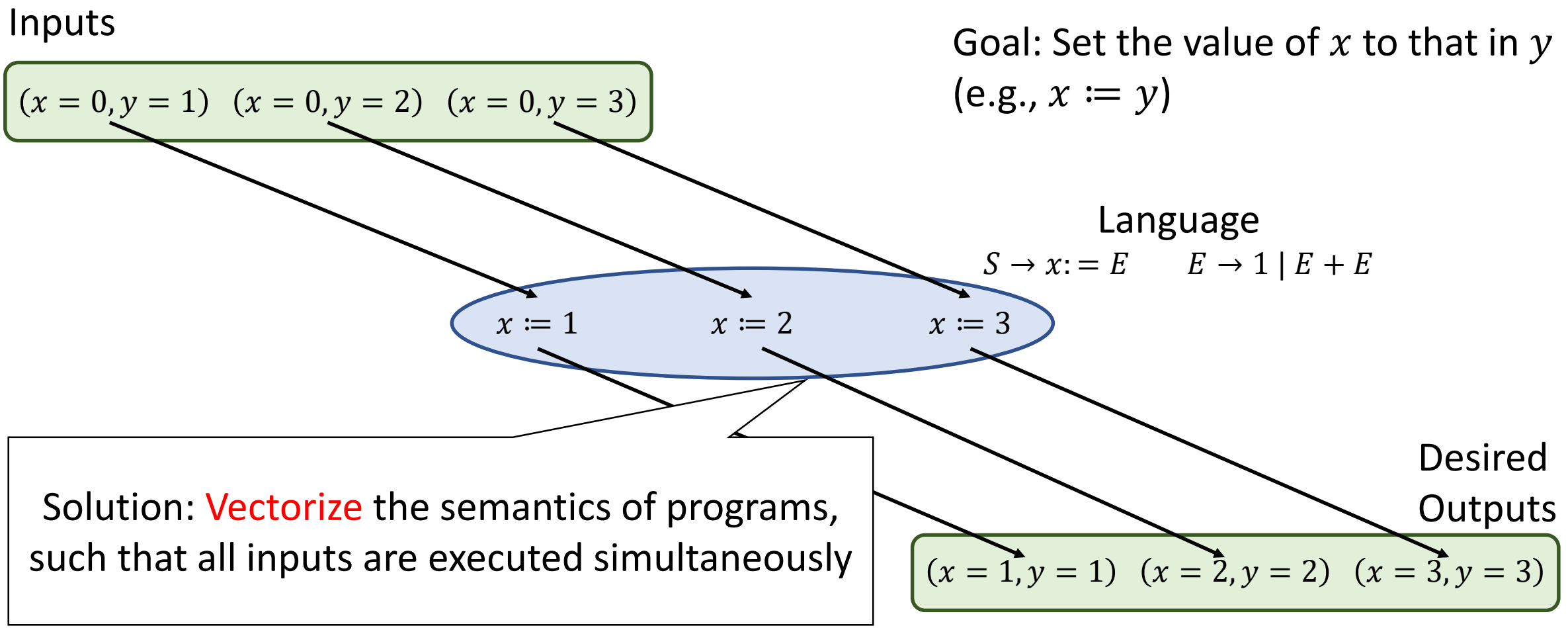
Synchronizing Between Inputs: Vectorized Semantics



Synchronizing Between Inputs: Vectorized Semantics



Synchronizing Between Inputs: Vectorized Semantics



Synchronizing Between Inputs: Vectorized Semantics

Inputs

$\langle (x = 0, y = 1), (x = 0, y = 2), (x = 0, y = 3) \rangle$

Goal: Set the value of x to that in y (e.g., $x := y$)

Language

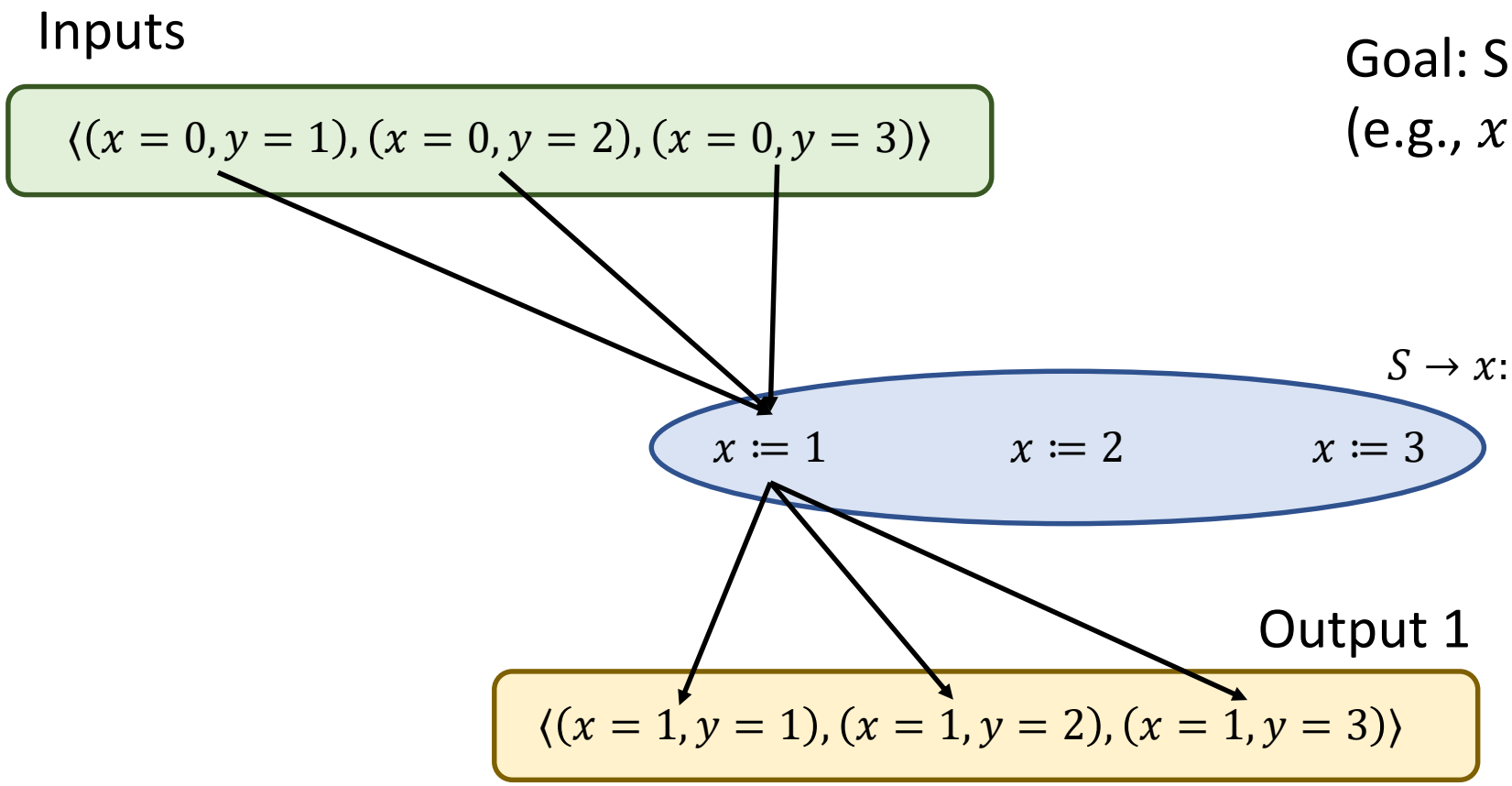
$S \to x := E \qquad E \to 1 \mid E + E$

$x := 1 \qquad x := 2 \qquad x := 3$

Desired Outputs

$\langle (x = 1, y = 1), (x = 2, y = 2), (x = 3, y = 3) \rangle$

Synchronizing Between Inputs: Vectorized Semantics

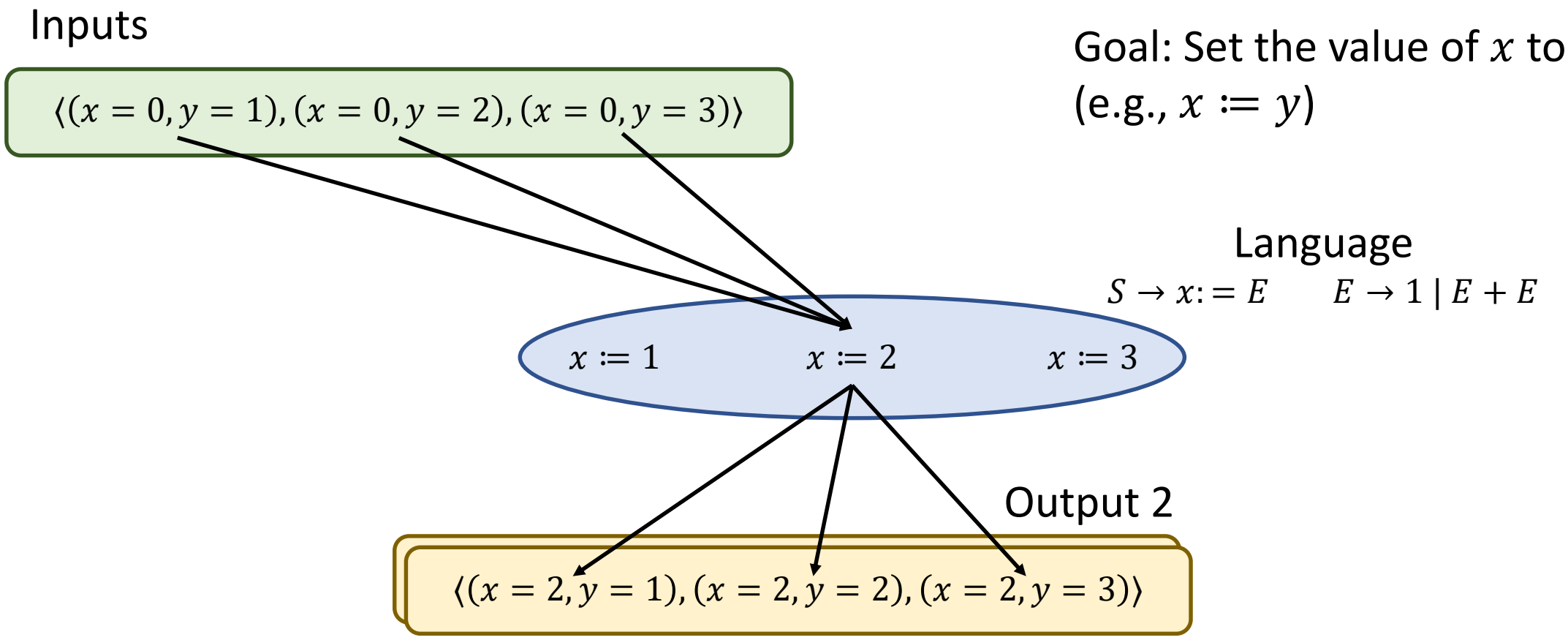


Goal: Set the value of x to that in y (e.g., $x := y$)

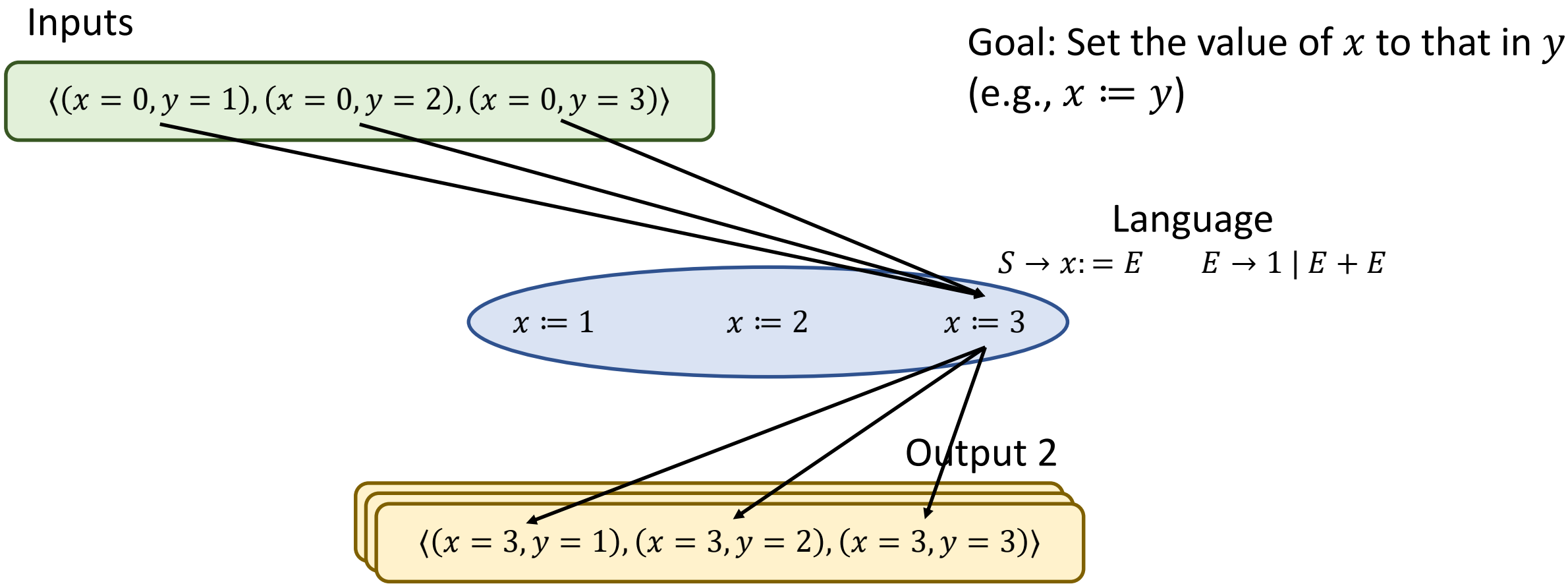
Language

$S \to x := E \qquad E \to 1 \mid E + E$

Synchronizing Between Inputs: Vectorized Semantics



Synchronizing Between Inputs: Vectorized Semantics



Synchronizing Between Inputs: Vectorized Semantics

Inputs

$\langle (x = 0, y = 1), (x = 0, y = 2), (x = 0, y = 3) \rangle$

Goal: Set the value of x to that in y (e.g., $x := y$)

Language

$S \to x := E \qquad E \to 1 \mid E + E$

$x := 1 \qquad x := 2 \qquad x := 3$

Output 3

$\langle (x = 3, y = 1), (x = 3, y = 2), (x = 3, y = 3) \rangle$

Set of outputs does not contain desired output

Synchronizing Between Inputs: Vectorized Semantics

Inputs

$\langle (x = 0, y = 1), (x = 0, y = 2), (x = 0, y = 3) \rangle$

Goal: Set the value of x to that in y (e.g., $x := y$)

Language

$S \to x := E \qquad E \to 1 \mid E + E$

$x := 1 \qquad x := 2 \qquad x := 3$

Output 3

$\langle (x = 3, y = 1), (x = 3, y = 2), (x = 3, y = 3) \rangle$

Thus, one can say that the problem is unrealizable!

Synchronizing Between Inputs: Vectorized Semantics

Inputs

$\langle (x = 0, y = 1), (x = 0, y = 2), (x = 0, y = 3) \rangle$

Goal: Set the value of x to that in y (e.g., $x := y$)

Definition:

$$[\![s]\!]\langle \sigma_1, \sigma_2, \cdots, \sigma_n \rangle = \langle [\![s]\!](\sigma_1), [\![s]\!](\sigma_2), \cdots, [\![s]\!](\sigma_n) \rangle$$

the
le!

$\langle (x = 3, y = 1), (x = 3, y = 2), (x = 3, y = 3) \rangle$

Putting it All Together

- Given a valid unrealizability triple $\{P\}\ S\ \{Q\}$…
- Where P and Q are predicates over the vectorized input and output state…
- If Q does not contain the (vectorized) desired output…
- Then the synthesis problem is unrealizable, as Q is an overapproximation of the reachable vectorized states!

Putting it All Together

- Given a valid unrealizability triple $\{P\}\ S\ \{Q\}$…

How can we obtain such valid triples?

Putting it All Together

- Given a valid unrealizability triple $\{P\}\ S\ \{Q\}$…

How can we obtain such valid triples?

By deriving them in an unrealizability logic proof tree!

Deriving a Triple: Proof by Structural Induction

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

Input: $x = 14, y = 15$ Desired Output: $1 \quad (14 \otimes 15)$

Deriving a Triple: Proof by Structural Induction

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

Input: $x = 14, y = 15$ Desired Output: $1 \quad (14 \otimes 15)$

Actual Output: e s.t. $e \,\&\, 4 = 4$

e is an auxiliary variable we use to indicate the result of expressions

Deriving a Triple: Proof by Structural Induction

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

Input: $x = 14, y = 15$

Desired Output: $1 \quad (14 \otimes 15)$

Actual Output: e s.t. $e \,\&\, 4 = 4$

$$\frac{}{\{x = 14 \wedge y = 15\}\ E\ \{e \,\&\, 4 = 1\}}\, HP$$

Deriving a Triple: Proof by Structural Induction

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

Input: $x = 14, y = 15$

Desired Output: $1 \quad (14 \otimes 15)$

Actual Output: e s.t. $e \,\&\, 4 = 4$

$$\frac{}{\underbrace{\{x = 14 \wedge y = 15\}}_{I} \; E \; \underbrace{\{e \,\&\, 4 = 1\}}_{O}} HP$$

Deriving a Triple: Proof by Structural Induction

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

Input: $x = 14, y = 15$

Desired Output: $1 \quad (14 \otimes 15)$

Actual Output: e s.t. $e \ \& \ 4 = 4$

$$\frac{}{\{x = 14 \wedge y = 15\} \ E \ \{e \ \& \ 4 = 1\}} \ HP$$

$$\Gamma$$

Introduce this as an induction hypothesis, into the context Γ!

Proof by Structural Induction

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

Input: $x = 14, y = 15$

Desired Output: $1 \quad (14 \otimes 15)$

Actual Output: e s.t. $e \ \& \ 4 = 4$

RHS Productions = Inductive cases

$$\frac{\Gamma \vdash \{I\}\, x\, \{O\} \quad \Gamma \vdash \{I\}\, y\, \{O\} \quad \boxed{\Gamma \vdash \{I\} E \ \& \ E \{O\} \quad \Gamma \vdash \{I\} E \mid E \{O\}}}{\{x = 14 \wedge y = 15\} \ E \ \{e \ \& \ 4 = 1\}} HP$$

Proof by Structural Induction

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

Input: $x = 14, y = 15$

Desired Output: $1 \quad (14 \otimes 15)$

Actual Output: e s.t. $e \,\&\, 4 = 4$

Base cases

$$\frac{\Gamma \vdash \{I\}\, x\, \{O\} \quad \Gamma \vdash \{I\}\, y\, \{O\} \quad \Gamma \vdash \{I\} E \,\&\, E \{O\} \quad \Gamma \vdash \{I\} E \mid E \{O\}}{\{x = 14 \wedge y = 15\}\ E\ \{e \,\&\, 4 = 1\}}\, HP$$

Proof by Structural Induction

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

Input: $x = 14, y = 15$

Desired Output: $1 \quad (14 \otimes 15)$

Actual Output: e s.t. $e \ \& \ 4 = 4$

$$\dfrac{\dfrac{}{\Gamma \vdash \{x = 14 \wedge y = 15\}\ x\ \{e = 14\}}\ Var}{\Gamma \vdash \{x = 14 \wedge y = 15\}\ x\ \{e \ \& \ 4 = 1\}}\ Weaken$$

$$\dfrac{\Gamma \vdash \{I\}\ x\ \{O\} \quad \Gamma \vdash \{I\}\ y\ \{O\} \quad \Gamma \vdash \{I\}E \ \& \ E\{O\} \quad \Gamma \vdash \{I\}E \mid E\{O\}}{\{x = 14 \wedge y = 15\}\ E\ \{e \ \& \ 4 = 1\}}\ HP$$

Proof by Structural Induction

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

Input: $x = 14, y = 15$

Desired Output: $1 \quad (14 \otimes 15)$

Actual Output: e s.t. $e \,\&\, 4 = 4$

$$\dfrac{\dfrac{}{\Gamma \vdash \{x = 14 \wedge y = 15\}\, y\, \{e = 15\}}\, Var}{\Gamma \vdash \{x = 14 \wedge y = 15\}\, y\, \{e \,\&\, 4 = 1\}}\, Weaken$$

$$\dfrac{\Gamma \vdash \{I\}\, x\, \{O\} \quad \Gamma \vdash \{I\}\, y\, \{O\} \quad \Gamma \vdash \{I\} E \,\&\, E \{O\} \quad \Gamma \vdash \{I\} E \mid E \{O\}}{\{x = 14 \wedge y = 15\}\ E\ \{e \,\&\, 4 = 1\}}\, HP$$

Proof by Structural Induction

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

Input: $x = 14, y = 15$

Desired Output: $1 \quad (14 \otimes 15)$

Actual Output: e s.t. $e \,\&\, 4 = 4$

Inductive cases

$$\frac{\Gamma \vdash \{I\}\, x\, \{O\} \quad \Gamma \vdash \{I\}\, y\, \{O\} \quad \Gamma \vdash \{I\}E \,\&\, E\{O\} \quad \Gamma \vdash \{I\}E \mid E\{O\}}{\{x = 14 \wedge y = 15\}\ E\ \{e \,\&\, 4 = 1\}}\, HP$$

Proof by Structural Induction

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

Input: $x = 14, y = 15$

Desired Output: $1 \quad (14 \otimes 15)$

Actual Output: e s.t. $e \,\&\, 4 = 4$

$$\frac{\dfrac{}{\Gamma \vdash \{x = 14 \wedge y = 15\}\, E\, \{e \,\&\, 4 = 1\}}\, IH \quad \dfrac{}{\Gamma \vdash \{x = 14 \wedge y = 15\}\, E\, \{e \& 4 = 1\}}\, IH}{\Gamma \vdash \{x = 14 \wedge y = 15\}\, E \,\&\, E\, \{e \,\&\, 4 = 1\}}\, And$$

$$\frac{\Gamma \vdash \{I\}\, x\, \{O\} \quad \Gamma \vdash \{I\}\, y\, \{O\} \quad \boxed{\Gamma \vdash \{I\} E \,\&\, E \{O\}} \quad \Gamma \vdash \{I\} E \mid E \{O\}}{\{x = 14 \wedge y = 15\} \;\; E \;\; \{e \,\&\, 4 = 1\}}\, HP$$

Proof by Structural Induction

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

Input: $x = 14, y = 15$

Desired Output: $1 \quad (14 \otimes 15)$

Actual Output: e s.t. $e \,\&\, 4 = 4$

$$\dfrac{\dfrac{}{\Gamma \vdash \{x = 14 \wedge y = 15\}\, E\, \{e \,\&\, 4 = 1\}}\, IH \quad \dfrac{}{\Gamma \vdash \{x = 14 \wedge y = 15\}\, E\, \{e \& 4 = 1\}}\, IH}{\Gamma \vdash \{x = 14 \wedge y = 15\}\, E \,\&\, E\, \{e \,\&\, 4 = 1\}}\, And$$

$$\dfrac{\Gamma \vdash \{I\}\, x\, \{O\} \quad \Gamma \vdash \{I\}\, y\, \{O\} \quad \Gamma \vdash \{I\} E \,\&\, E \{O\} \quad \Gamma \vdash \{I\} E \mid E \{O\}}{\{x = 14 \wedge y = 15\} \;\; E \;\; \{e \,\&\, 4 = 1\}}\, HP$$

Proof by Structural Induction

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

Input: $x = 14, y = 15$

Desired Output: $1 \quad (14 \otimes 15)$

Actual Output: e s.t. $e \,\&\, 4 = 4$

$$\frac{\dfrac{}{\Gamma \vdash \{x = 14 \wedge y = 15\}\, E\, \{e \,\&\, 4 = 1\}}\, IH \qquad \dfrac{}{\Gamma \vdash \{x = 14 \wedge y = 15\}\, E\, \{e \& 4 = 1\}}\, IH}{\Gamma \vdash \{x = 14 \wedge y = 15\}\, E \mid E\, \{e \,\&\, 4 = 1\}}\, Or$$

$$\frac{\Gamma \vdash \{I\}\, x\, \{O\} \quad \Gamma \vdash \{I\}\, y\, \{O\} \quad \Gamma \vdash \{I\} E \,\&\, E \{O\} \quad \boxed{\Gamma \vdash \{I\} E \mid E \{O\}}}{\{x = 14 \wedge y = 15\} \;\; E \;\; \{e \,\&\, 4 = 1\}}\, HP$$

Proof by Structural Induction

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

Input: $x = 14, y = 15$

Desired Output: $1 \quad (14 \otimes 15)$

Actual Output: e s.t. $e \,\&\, 4 = 4$

$$\frac{\overset{\cdots}{\Gamma \vdash \{I\}\, x\, \{O\}} \quad \overset{\cdots}{\Gamma \vdash \{I\}\, y\, \{O\}} \quad \overset{\cdots}{\Gamma \vdash \{I\} E \,\&\, E \{O\}} \quad \overset{\cdots}{\Gamma \vdash \{I\} E \mid E \{O\}}}{\{x = 14 \wedge y = 15\}\ E\ \{e \,\&\, 4 = 1\}}\, HP$$

Can prove this is true in a Hoare-like manner!

Proof by Structural Induction

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

Input: $x = 14, y = 15$

Desired Output: $1 \quad (14 \otimes 15)$

Actual Output: e s.t. $e \,\&\, 4 = 4$

$\{x = 14 \wedge y = 15\} \;\; E \;\; \{e \,\&\, 4 = 1\}$:

- Starting from the input $x = 14 \wedge y = 15$,
- The result $e \,\&\, 4 = 1$ is an *overapproximation*

Proof by Structural Induction

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

Input: $x = 14, y = 15$

Desired Output: $1 \quad (14 \otimes 15)$

Actual Output: e s.t. $e \ \& \ 4 = 4$

$\{x = 14 \wedge y = 15\} \ E \ \{e \ \& \ 4 = 1\}$:

- Starting from the input $x = 14 \wedge y = 15$,
- The result $e \ \& \ 4 = 1$ is an *overapproximation*

- Note $e \ \& \ 4 = 1 \cap e = 1 \equiv \phi$
- Thus, the problem is *unrealizable*

Paper Highlights

- Unrealizability logic is both sound and relatively complete (even for grammars containing loops!)

- Vectorized semantics extend naturally to infinite vectors as well, allowing us to prove unrealizability for problems requiring an infinite number of examples

- Other examples and theoretical results such as decidability under certain fragments

Questions?

Inference Rules in Unrealizability Logic

Inference Rules in Unrealizability Logic

- In addition to the two aspects explained before, unrealizability logic poses a unique challenge when composing values for productions such as $E \rightarrow E + E$

UL with Program Synchronization

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]$
- Output: Should be $[4,6]$

UL with Program Synchronization

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $\underbrace{[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]}$ Call this P
- Output: Should be $[4,6]$

UL with Program Synchronization

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $\underbrace{[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]}$ Call this P
- Output: Should be $[4,6]$

$$\frac{\dots \quad \{P\}\ E_1\ \{e_t = [1,2] \vee [3,4]\} \qquad \dots \quad \{P\}\ E_2\ \{e_t = [3,4] \vee [1,2]\}}{\{P\}\ E_1 + E_2\ \{???\}}\ Plus?$$

UL with Program Synchronization

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $\underbrace{[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]}$ Call this P
- Output: Should be $[4,6]$

These two premises: clearly correct

$$\frac{\{P\}\ E_1\ \{e_t = [1,2] \vee [3,4]\} \qquad \{P\}\ E_2\ \{e_t = [3,4] \vee [1,2]\}}{\{P\}\ E_1 + E_2\ \{???\}}\ Plus?$$

UL with Program Synchronization

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $\underbrace{[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]}$ Call this P
- Output: Should be $[4,6]$

$$\frac{\dots \quad \dots}{\{P\}\ E_1\ \{e_t = [1,2] \vee [3,4]\} \quad \{P\}\ E_2\ \{e_t = [3,4] \vee [1,2]\}}{\{P\}\ E_1 + E_2\ \{???\}}\ Plus?$$

UL with Program Synchronization

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $\underbrace{[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]}_{\text{Call this } P}$
- Output: Should be $[4,6]$

$$\frac{\dots \quad \dots}{\{P\}\ E_1\ \{e_t = [1,2] \vee [3,4]\} \quad \{P\}\ E_2\ \{e_t = [3,4] \vee [1,2]\}}\ Plus?$$
$$\{P\}\ E_1 + E_2\ \{???\}$$

UL with Program Synchronization

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $\underbrace{[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]}$ Call this P
- Output: Should be $[4,6]$

$$\frac{\dots \quad \dots}{\{P\}\ E_1\ \{e_t = [1,2] \vee [3,4]\} \quad \{P\}\ E_2\ \{e_t = [3,4] \vee [1,2]\}}{\{P\}\ E_1 + E_2\ \{???\}}\ Plus?$$

UL with Program Synchronization

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $\underbrace{[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]}$ Call this P
- Output: Should be $[4,6]$

$$\frac{\dots \quad \{P\}\ E_1\ \{e_t = [1,2] \vee [3,4]\} \qquad \dots \quad \{P\}\ E_2\ \{e_t = [3,4] \vee [1,2]\}}{\{P\}\ E_1 + E_2\ \{???\}}\ Plus?$$

UL with Program Synchronization

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $\underbrace{[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]}$ Call this P
- Output: Should be $[4,6]$

Filling the ? ? ? in: Harder!

$$\frac{\dots \qquad \dots}{\{P\}\ E_1\ \{e_t = [1,2] \vee [3,4]\} \quad \{P\}\ E_2\ \{e_t = [3,4] \vee [1,2]\}}\ Plus?$$
$$\{P\}\ E_1 + E_2\ \boxed{\{???\}}$$

UL with Program Synchronization

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $\underbrace{[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]}$ Call this P
- Output: Should be $[4,6]$

Filling the ? ? ? in: Harder!
Naïve approach gives us
$[2,4], [4,6], [6,8]$

$$\frac{\dots \qquad \dots}{\{P\}\ E_1\ \{e_t = [1,2] \vee [3,4]\} \quad \{P\}\ E_2\ \{e_t = [3,4] \vee [1,2]\}}\ Plus?$$
$$\{P\}\ E_1 + E_2\ \boxed{\{???\}}$$

UL with Program Synchronization

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $\underbrace{[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]}$ Call this P
- Output: Should be $[4,6]$

Filling the ??? in: Harder!
Naïve approach gives us
$[2,4]$, $[4,6]$, $[6,8]$

$$\frac{\dots \qquad \dots}{\{P\}\ E_1\ \{e_t = [1,2] \vee [3,4]\} \qquad \{P\}\ E_2\ \{e_t = [3,4] \vee [1,2]\}}\ Plus?$$
$$\{P\}\ E_1 + E_2\ \{???\}$$

UL with Program Synchronization

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $\underbrace{[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]}$ Call this P
- Output: Should be $[4,6]$

Filling the ??? in: Harder!
Naïve approach gives us
$[2,4], [4,6], [6,8]$

$$\frac{\dots \qquad \dots}{\{P\}\ E_1\ \{e_t = [1,2] \vee [3,4]\} \qquad \{P\}\ E_2\ \{e_t = [3,4] \vee [1,2]\}}{\{P\}\ E_1 + E_2\ \{???\}}\ Plus?$$

UL with Program Synchronization

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $\underbrace{[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]}$ Call this P
- Output: Should be $[4,6]$

Filling the ??? in: Harder!
Naïve approach gives us
$[2,4], [4,6], [6,8]$

$$\frac{\dots \qquad \dots}{\{P\}\ E_1\ \{e_t = [1,2] \vee [3,4]\} \quad \{P\}\ E_2\ \{e_t = [3,4] \vee [1,2]\}}\ Plus?$$
$$\{P\}\ E_1 + E_2\ \{???\}$$

UL with Program Synchronization

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $\underbrace{[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]}$ Call this P
- Output: Should be $[4,6]$

Filling the ??? in: Harder!
Naïve approach gives us
$[2,4], [4,6], \boxed{[6,8]}$

$$\frac{\dots \qquad \dots}{\{P\}\ E_1\ \{e_t = [1,2] \vee \boxed{[3,4]}\} \qquad \{P\}\ E_2\ \{e_t = \boxed{[3,4]} \vee [1,2]\}}\ Plus?$$
$$\{P\}\ E_1 + E_2\ \{???\}$$

UL with Program Synchronization

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]$
- Output: Should be $[4,6]$

These two entries come from *different* states in the precondition.

However, they were added together in the naïve version!

$$\frac{\dots \quad \dots}{\{P\}\ E_1\ \{e_t = [1,2] \vee [3,4]\} \quad \{P\}\ E_2\ \{e_t = [3,4] \vee [1,2]\}}\ Plus?$$
$$\{P\}\ E_1 + E_2\ \{???\}$$

UL with Program Synchronization

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]$
- Output: Should be $[4,6]$

Problem: Failing to consider that only results from the same input-state should be composed

$$\frac{\dots \quad \dots}{\{P\}\ E_1\ \{e_t = [1,2] \vee [3,4]\} \quad \{P\}\ E_2\ \{e_t = [3,4] \vee [1,2]\}}{\{P\}\ E_1 + E_2\ \{???\}}\ Plus?$$

UL with Program Synchronization: Ghost State

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]$

- Solution: Extend the input states with *ghost state*

UL with Program Synchronization: Ghost State

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]$

- $[(x_1, y_1, x_1', y_1'), (x_2, y_2, x_2', y_2')] =$
  $[(1,3,1,3), (2,4,2,4)], [(3,1,3,1), (4,2,4,2)]$

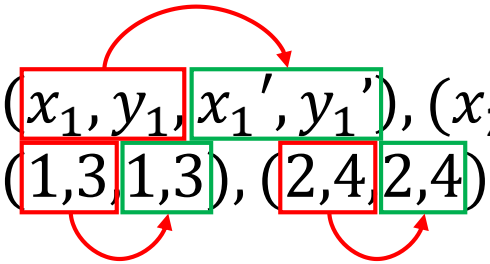
- Solution: Extend the input states with *ghost state*

- Ghost state: Simply a copy of the original state, with *fresh* variables

UL with Program Synchronization: Ghost State

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]$



- $[(x_1, y_1, x_1', y_1'), (x_2, y_2, x_2', y_2')] =$
  $[(1,3,1,3), (2,4,2,4)], [(3,1,3,1), (4,2,4,2)]$

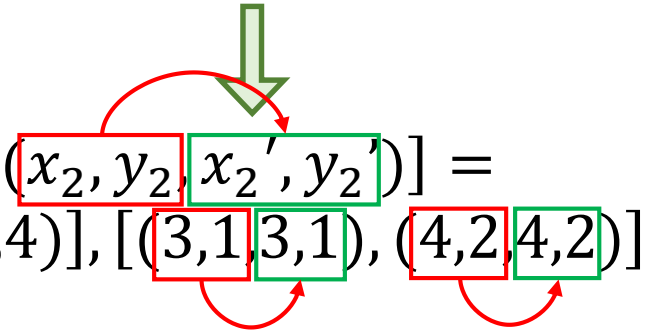
- Solution: Extend the input states with *ghost state*

- Ghost state: Simply a copy of the original state, with *fresh* variables

UL with Program Synchronization: Ghost State

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]$
- $[(x_1, y_1, x_1', y_1'), (x_2, y_2, x_2', y_2')] =$
  $[(1,3,1,3), (2,4,2,4)], [(3,1,3,1), (4,2,4,2)]$



- Solution: Extend the input states with *ghost state*
- Ghost state: Simply a copy of the original state, with *fresh* variables

UL with Program Synchronization: Ghost State

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]$

- $[(x_1, y_1, x_1', y_1'), (x_2, y_2, x_2', y_2')] =$
  $[(1,3,1,3), (2,4,2,4)], [(3,1,3,1), (4,2,4,2)]$

- Solution: Extend the input states with *ghost state*
- Ghost state: Simply a copy of the original state, with *fresh* variables
- Point: Primed variables are not ‘executed’, so they can track the original state!

UL with Program Synchronization: Ghost State

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]$
- $[(x_1, y_1, x_1', y_1'), (x_2, y_2, x_2', y_2')] =$ $[(1,3,1,3), (2,4,2,4)], [(3,1,3,1), (4,2,4,2)]$ } Call this P_{ext}

$$\frac{\dots \qquad \dots}{\{P_{ext}\}\ E_1\ \{e_t = [1,2] \vee [3,4] \dots\} \qquad \{P_{ext}\}\ E_2\ \{e_t = [3,4] \vee [1,2] \dots\}}\ Plus?$$
$$\{P\}\ E_1 + E_2\ \{???\}$$

UL with Program Synchronization: Ghost State

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]$
- $[(x_1, y_1, x_1', y_1'), (x_2, y_2, x_2', y_2')] = [(1,3,1,3), (2,4,2,4)], [(3,1,3,1), (4,2,4,2)]$ } Call this P_{ext}

$(e_t = [1,2] \wedge (x_1, y_1, x_1', y_1') = (1,3,1,3) \wedge (x_2, y_2, x_2', y_2') = (2,4,2,4)) \vee$
$(e_t = [3,4] \wedge (x_1, y_1, x_1', y_1') = (3,1,3,1) \wedge (x_2, y_2, x_2', y_2') = (4,2,4,2))$

$$\frac{\{P_{ext}\}\ E_1\ \{e_t = [1,2] \vee [3,4] \ldots\} \quad \ldots \quad \{P_{ext}\}\ E_2\ \{e_t = [3,4] \vee [1,2] \ldots\}}{\{P\}\ E_1 + E_2\ \{???\}}\ Plus?$$

UL with Program Synchronization: Ghost State

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Input: $[(x_1, y_1), (x_2, y_2)] = [(1,3), (2,4)], [(3,1), (4,2)]$

$(e_t = [3,4] \wedge (x_1, y_1, x_1', y_1') = (1,3,1,3) \wedge (x_2, y_2, x_2', y_2') = (2,4,2,4)) \vee$
$(e_t = [1,2] \wedge (x_1, y_1, x_1', y_1') = (3,1,3,1) \wedge (x_2, y_2, x_2', y_2') = (4,2,4,2))$

$$\frac{\dots \qquad \dots}{\{P_{ext}\}\ E_1\ \{e_t = [1,2] \vee [3,4] \dots\} \qquad \{P_{ext}\}\ E_2\ \{e_t = [3,4] \vee [1,2] \dots\}}\ Plus?$$
$$\{P\}\ E_1 + E_2\ \{???\}$$

UL with Program Synchronization: Ghost State

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Compose only when ghost state matches:
  Green ghost states match, thus [3,4] and [1,2] are composed to produce [6,8]

$$(e_t = [3,4] \wedge (x_1, y_1, x_1', y_1') = (1,3,1,3) \wedge (x_2, y_2, x_2', y_2') = (2,4,2,4)) \vee (e_t = [1,2] \wedge (x_1, y_1, x_1', y_1') = (3,1,3,1) \wedge (x_2, y_2, x_2', y_2') = (4,2,4,2))$$

$$(e_t = [1,2] \wedge (x_1, y_1, x_1', y_1') = (1,3,1,3) \wedge (x_2, y_2, x_2', y_2') = (2,4,2,4)) \vee (e_t = [3,4] \wedge (x_1, y_1, x_1', y_1') = (3,1,3,1) \wedge (x_2, y_2, x_2', y_2') = (4,2,4,2))$$

$$\frac{\{P_{ext}\}\ E_1\ \{e_t = [1,2] \vee [3,4] \ldots\} \quad \ldots \quad \{P_{ext}\}\ E_2\ \{e_t = [3,4] \vee [1,2] \ldots\}}{\{P\}\ E_1 + E_2\ \{???\}}\ Plus?$$

UL with Program Synchronization: Ghost State

$E ::= E_1 + E_2$
$E_1 ::= x$
$E_2 ::= y$

- Compose only when ghost state matches:
  Red ghost states do *not* match, thus [3,4] and [3,4] are *not* composed in the final postcondition!

$$(e_t = [3,4] \wedge (x_1, y_1, x_1', y_1') = (1,3,1,3) \wedge (x_2, y_2, x_2', y_2') = (2,4,2,4)) \vee$$
$$(e_t = [1,2] \wedge (x_1, y_1, x_1', y_1') = (3,1,3,1) \wedge (x_2, y_2, x_2', y_2') = (4,2,4,2))$$

$$(e_t = [1,2] \wedge (x_1, y_1, x_1', y_1') = (1,3,1,3) \wedge (x_2, y_2, x_2', y_2') = (2,4,2,4)) \vee$$
$$(e_t = [3,4] \wedge (x_1, y_1, x_1', y_1') = (3,1,3,1) \wedge (x_2, y_2, x_2', y_2') = (4,2,4,2))$$

$$\frac{\{P_{ext}\}\ E_1\ \{e_t = [1,2] \vee [3,4] \ldots\} \quad \ldots \quad \{P_{ext}\}\ E_2\ \{e_t = [3,4] \vee [1,2] \ldots\}}{\{P\}\ E_1 + E_2\ \{???\}}\ Plus?$$

UL with Program Synchronization: Ghost State

$$E ::= E_1 + E_2$$
$$E_1 ::= x$$
$$E_2 ::= y$$

- Compose only when ghost state matches:
  Red ghost states do *not* match, thus $[3,4]$ and $[3,4]$ are *not* composed in the final postcondition!
- Thus final postcondition only contains $\mathrm{e_t} = [6,8]$ as desired

$$\frac{\dots \qquad \dots}{\{P_{ext}\}\ E_1\ \{e_t = [1,2] \vee [3,4] \dots\} \qquad \{P_{ext}\}\ E_2\ \{e_t = [3,4] \vee [1,2] \dots\}}\ Plus?$$
$$\{P\}\ E_1 + E_2\ \{\mathrm{e_t} = [6,8] \dots\}$$

Plus as a Rule

$$\frac{\{P \wedge v = v_1\}E_1\{Q_1\} \qquad \{P \wedge v = v_2\}E_2\{Q_2\}}{\{P\}\, E_1 + E_2\, \{\exists e_t', v_1, v_2, v_1', v_2'.\ (P \wedge Q_1[v_1`\backslash v] \wedge Q_2[v_2'\backslash v])[e_t`\backslash e_t] \wedge e_t = e_{t_1}` + e_{t_2}`}$$

Plus as a Rule

Generates ghost state for E_1, using (fresh) vars v_1

$$\frac{\{P \wedge v = v_1\}E_1\{Q_1\} \qquad \{P \wedge v = v_2\}E_2\{Q_2\}}{\{P\}\ E_1 + E_2\ \{\exists e_t', v_1, v_2, v_1', v_2'.\ (P \wedge Q_1[v_1`\backslash v] \wedge Q_2[v_2'\backslash v])[e_t`\backslash e_t] \wedge e_t = e_{t_1}` + e_{t_2}`}$$

Plus as a Rule

Generates ghost state for E_2, using (fresh) vars v_2

$$\frac{\{P \wedge v = v_1\}E_1\{Q_1\} \qquad \{P \wedge v = v_2\}E_2\{Q_2\}}{\{P\}\ E_1 + E_2\ \{\exists e_t', v_1, v_2, v_1', v_2'.\ (P \wedge Q_1[v_1`\backslash v] \wedge Q_2[v_2'\backslash v])[e_t`\backslash e_t] \wedge e_t = e_{t_1}` + e_{t_2}`}$$

Plus as a Rule

$$\frac{\{P \wedge v = v_1\}E_1\{Q_1\} \qquad \{P \wedge v = v_2\}E_2\{Q_2\}}{\{P\}\, E_1 + E_2\, \{\exists e_t', v_1, v_2, v_1', v_2'.\ (P \wedge Q_1[v_1`\backslash v] \wedge Q_2[v_2'\backslash v])[e_t`\backslash e_t] \wedge e_t = e_{t_1}` + e_{t_2}`}$$

Checks that ghost states match:
note that predicate only contains
‘extended‘ states where ghost states match!

Plus as a Rule

$$\frac{\{P \wedge v = v_1\}E_1\{Q_1\} \qquad \{P \wedge v = v_2\}E_2\{Q_2\}}{\{P\}\ E_1 + E_2\ \{\exists e_t', v_1, v_2, v_1', v_2'.\ (P \wedge Q_1[v_1`\backslash v] \wedge Q_2[v_2'\backslash v])[e_t`\backslash e_t] \wedge e_t = e_{t_1}` + e_{t_2}'}$$

Updates e_t to the value computed by +

Plus as a Rule

$$\frac{\{P \wedge v = v_1\}E_1\{Q_1\} \qquad \{P \wedge v = v_2\}E_2\{Q_2\}}{\{P\}\, E_1 + E_2\, \{\exists e_t', v_1, v_2, v_1', v_2'.\ (P \wedge Q_1[v_1`\backslash v] \wedge Q_2[v_2'\backslash v])[e_t`\backslash e_t] \wedge e_t = e_{t_1}` + e_{t_2}`\}}$$

Quantifies additionally introduced variables out

Plus as a Rule

$$\frac{\{P \wedge v = v_1\}E_1\{Q_1\} \qquad \{P \wedge v = v_2\}E_2\{Q_2\}}{\{P\}\, E_1 + E_2\, \{\exists e_t', v_1, v_2, v_1', v_2'.\ (P \wedge Q_1[v_1`\backslash v] \wedge Q_2[v_2'\backslash v])[e_t`\backslash e_t] \wedge e_t = e_{t_1}` + e_{t_2}`}$$

- Complete rules for all program constructs can be developed using the idea of ghost state
- Thus, unrealizability logic is *relatively complete* (even for loops)!

Questions?

A Surprising Similarity

- Unrealizability logic is very similar to Hoare logic with recursion and nondeterminism

- In fact, synthesis itself can be modeled using nondeterminism

Modeling Synthesis using Nondeterminism

$$E ::= 0 \mid E + 1 \mid E - 1$$

Modeling Synthesis using Nondeterminism

$$E ::= 0 \mid E + 1 \mid E - 1$$

Can interpret this as a nondeterministic, recursive program

Modeling Synthesis using Nondeterminism

$$E ::= 0 \mid E + 1 \mid E - 1$$

Can interpret this as a nondeterministic, recursive program

```
E(){
  if (*) return 0;
  else if (*) return E() + 1;
  else return E() – 1;
}
main(){
  r = E();
  assert (!spec(r));
}
```

Modeling Synthesis using Nondeterminism

$$E ::= 0 \mid E + 1 \mid E - 1$$

Can interpret this as a nondeterministic, recursive program

```
E(){
  if (*) return 0;
  else if (*) return E() + 1;
  else return E() – 1;
}
main(){
  r = E();
  assert (!spec(r));
}
```

- If verification works on this example (using standard overapproximating techniques): $!\mathrm{spec}(\mathrm{r})$ is guaranteed to hold.
- Thus the problem is *unrealizable*.

Modeling Synthesis using Nondeterminism

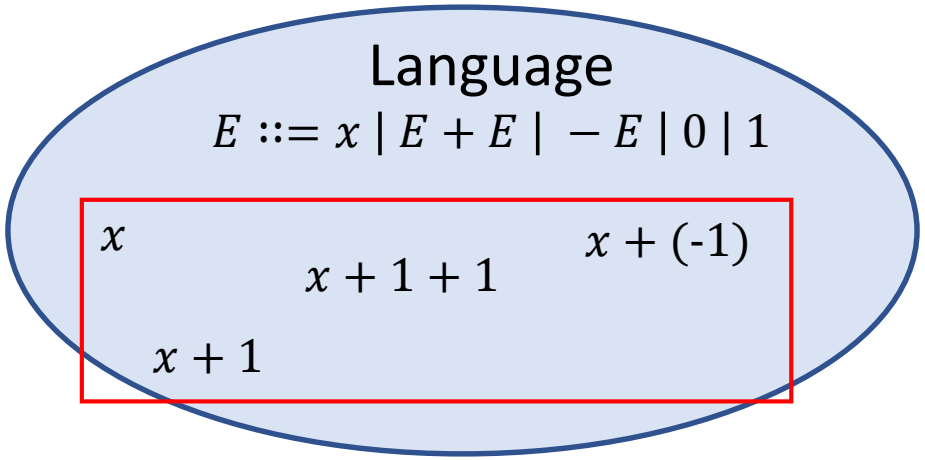
$$E ::= 0 \mid E + 1 \mid E - 1$$

Can interpret this as a nondeterministic, recursive program

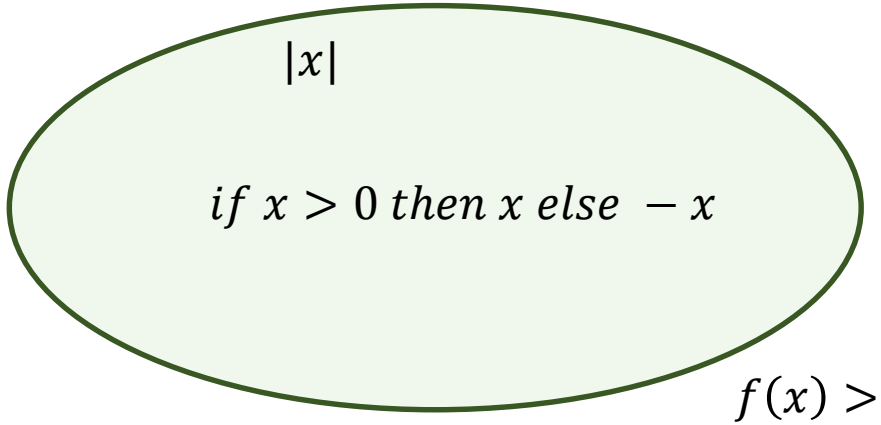
```
E(){
  if (*) return 0;
  else if (*) return E() + 1;
  else return E() – 1;
}
main(){
  r = E();
  assert (!spec(r));
}
```

- If a bug catcher can *prove* that the assertion can be violated (using underapproximating techniques), there exists a path s.t. $\mathrm{spec}(\mathrm{r})$ holds.
- Thus the problem *has a solution*.

To Prove Unrealizability…



- We need to characterize the behavior of a *set* of programs
- i.e., does the set of programs in E simultaneously not satisfy the specification?



Behavioral Spec

$f(x) > 0 \wedge (f(x) = x \vee f(x) = -x)$

The Challenges of Unrealizability Logic

- Because we are dealing with sets of programs and sets of examples, we need a way to *synchronize* between these aspects

- Two main synchronization obstacles: i) Examples, ii) Programs

UL with Multiple Examples

- Multiple examples in UL require *synchronization*

UL with Multiple Examples

$E ::= 0 \mid E + 1 \mid E - 1$

- Input: $x \in \mathbb{Z}$
- Output: e s.t. $e = x$
- Clearly unrealizable!

UL with Multiple Examples

$E ::= 0 \mid E + 1 \mid E - 1$

- Input: $x \in \mathbb{Z}$
- Output: e s.t. $e = x$
- Clearly unrealizable!

$$\frac{\dots}{\dfrac{\{x = x_{aux}\}\ E\ \{x = x_{aux} \wedge \exists k.\, e_t = k\}}{\{x = x_{aux}\}\ E\ \{e_t \neq x_{aux}\}}\, Weaken?}$$

UL with Multiple Examples

$E \ ::= 0 \mid E + 1 \mid E - 1$

- Input: $x \in \mathbb{Z}$
- Output: e s.t. $e = x$
- Clearly unrealizable!

This part is clearly true:
E can generate any fixed integer

$$\frac{\dfrac{\dots}{\{x = x_{aux}\}\ E\ \{x = x_{aux} \wedge \exists k.\, e_t = k\}}}{\{x = x_{aux}\}\ E\ \{e_t \neq x_{aux}\}}\ Weaken?$$

UL with Multiple Examples

$E ::= 0 \mid E + 1 \mid E - 1$

- Input: $x \in \mathbb{Z}$
- Output: e s.t. $e = x$
- Clearly unrealizable!

This part is NOT:
Clearly k can also be x_{aux}!

$$\frac{\dots \quad \{x = x_{aux}\}\ E\ \{x = x_{aux} \wedge \exists k.\, e_t = k\}}{\{x = x_{aux}\}\ E\ \{e_t \neq x_{aux}\}}\ Weaken?$$

UL with Multiple Examples

$E ::= 0 \mid E + 1 \mid E - 1$

- Input: $x \in \mathbb{Z}$
- Output: e s.t. $e = x$
- Clearly unrealizable!

The problem:
Fails to consider that there must be a *single* term for each x_{aux} (example)

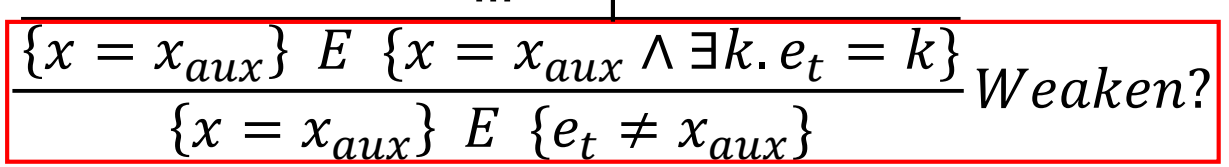
$$\frac{\dots}{\dfrac{\{x = x_{aux}\}\ E\ \{x = x_{aux} \wedge \exists k.\, e_t = k\}}{\{x = x_{aux}\}\ E\ \{e_t \neq x_{aux}\}}\ Weaken?}$$

UL with Multiple Examples

$E ::= 0 \mid E + 1 \mid E - 1$

- Input: $x \in \mathbb{Z}$
- Output: e s.t. $e = x$
- Clearly unrealizable!

Solution: Synchronize the examples



UL with Multiple Examples: Vectorization

$E ::= 0 \mid E + 1 \mid E - 1$

- Input: $x \in \mathbb{Z}$
- Output: e s.t. $e = x$
- Clearly unrealizable!

Also clearly true:
E will generate the same integer k for *every* i

$$\frac{\dfrac{\dots}{\{\forall i.\, x_i = i\}\ E\ \{\exists k.\, \forall i.\, x_i = i \wedge e_{t_i} = k\}}}{\{\forall i.\, x_i = i\}\ E\ \{\exists i.\, e_{t_i} \neq i\}}\ Weaken?$$

UL with Multiple Examples: Vectorization

$E ::= 0 \mid E + 1 \mid E - 1$

- Input: $x \in \mathbb{Z}$
- Output: e s.t. $e = x$
- Clearly unrealizable!

True this time:
There will exist some i such that $i \neq k$

$$\frac{\ldots}{\dfrac{\{\forall i.\, x_i = i\}\ E\ \{\exists k.\, \forall i.\, x_i = i \wedge e_{t_i} = k\}}{\{\forall i.\, x_i = i\}\ E\ \{\exists i.\, e_{t_i} \neq i\}}\ Weaken?}$$

UL with Multiple Examples: Vectorization

$E ::= 0 \mid E + 1 \mid E - 1$

- Input: $x \in \mathbb{Z}$
- Output: e s.t. $e = x$
- Clearly unrealizable!

In essence:
We have circumvented the problem by analyzing examples *simultaneously,* making sure they follow the same path
(Using an alternative program semantics)

$$\frac{\dots}{\{\forall i.\, x_i = i\}\ E\ \{\exists k.\, \forall i.\, x_i = i \wedge e_{t_i} = k\}}\ Weaken?$$
$$\{\forall i.\, x_i = i\}\ E\ \{\exists i.\, e_{t_i} \neq i\}$$