SIMBA: Inductive Program Synthesis via Iterative Forward-Backward Abstract Interpretation

Yongho Yoon, Seoul National University & Sparrow Co., Ltd.
Woosuk Lee, Hanyang University
Kwangkeun Yi, Seoul National University

20/06/2023 @ PLDI 2023

PLDI Orlando 2023

Sparrow

Inductive Program Synthesis

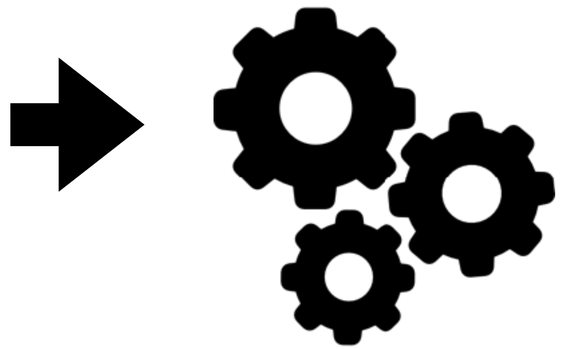
in SyGuS Format

Syntactic Spec

$$
\begin{array}{rcl}
S & \rightarrow & x \mid 0001_2 \\
 & \mid & S \wedge S \mid S \vee S \mid S \oplus S \\
 & \mid & S + S \mid S \times S \mid S / S \mid S >> S
\end{array}
$$

Semantic Spec

$$f(1011_2) = 0011_2$$



$$f(x) = ((x + 0001_2) \oplus x) >> 0001_2$$

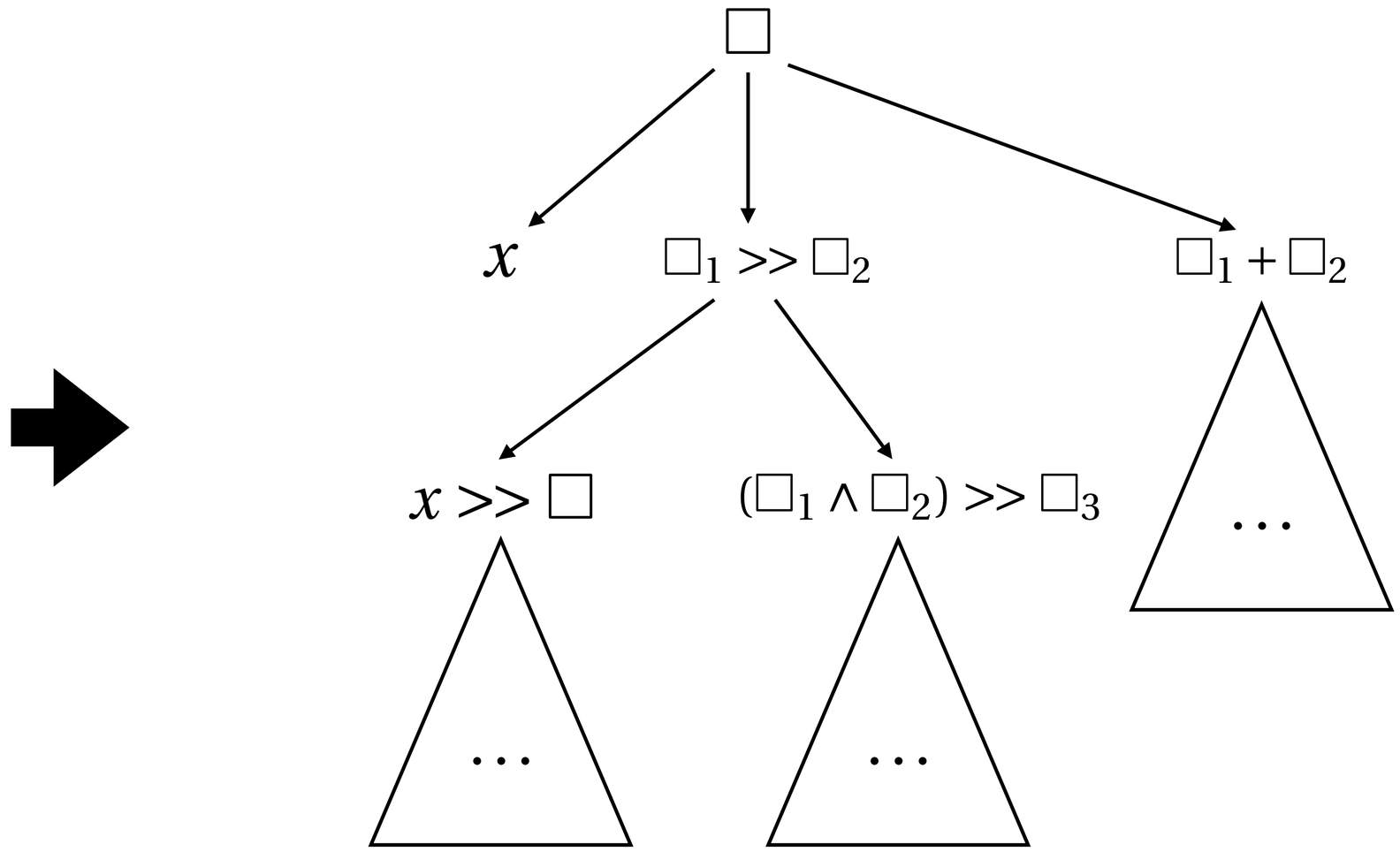
Pruning by Feasibility

Syntactic Spec

$$
\begin{aligned}
S \quad &\rightarrow \quad x \mid 0001_2 \\
&\mid \quad S \wedge S \mid S \vee S \mid S \oplus S \\
&\mid \quad S + S \mid S \times S \mid S / S \mid S >> S
\end{aligned}
$$

Semantic Spec

$f(1011_2) = 0011_2$



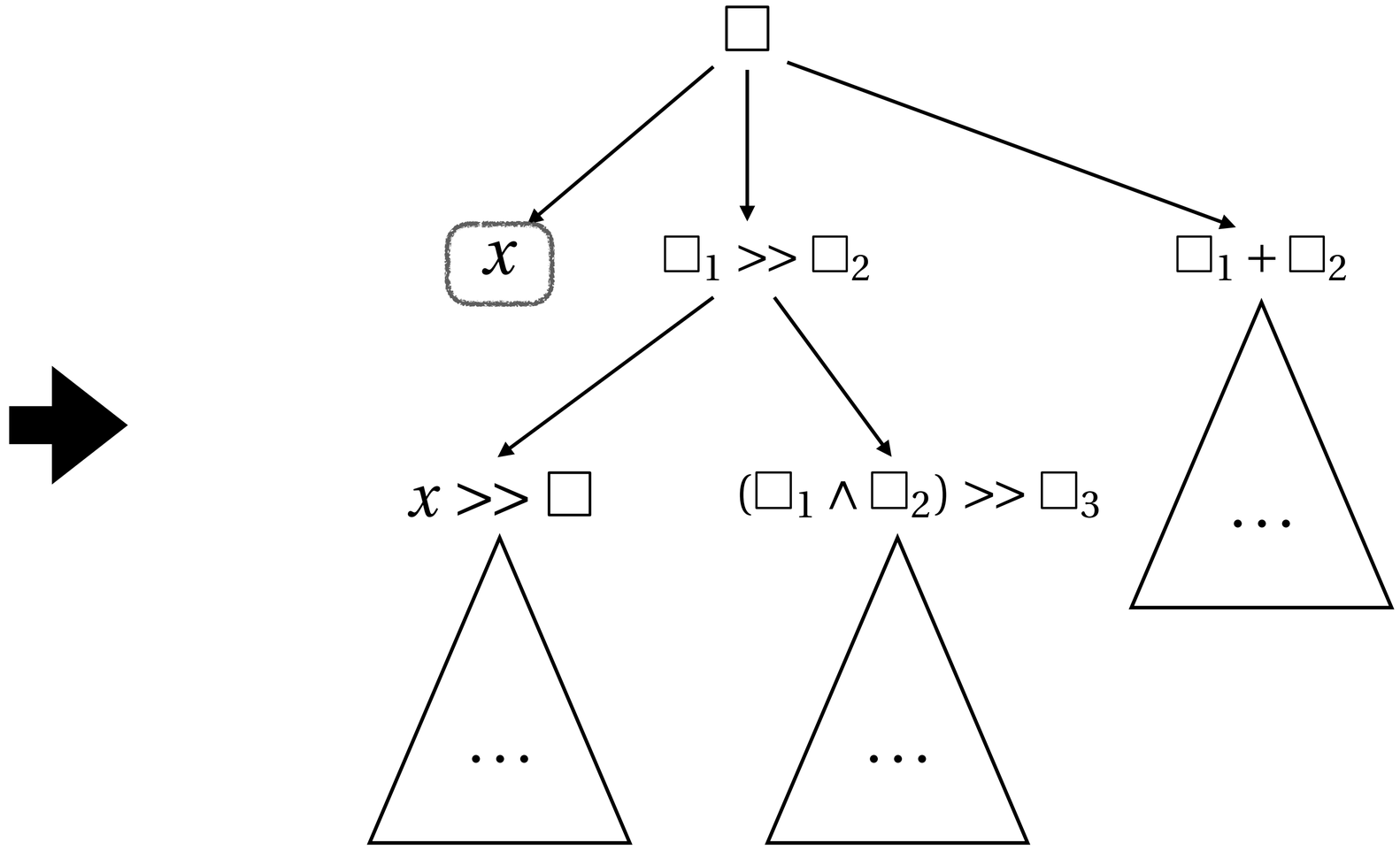
Pruning by Feasibility

Syntactic Spec

$$
\begin{aligned}
S \quad &\rightarrow \quad x \mid 0001_2 \\
&\mid \quad S \wedge S \mid S \vee S \mid S \oplus S \\
&\mid \quad S + S \mid S \times S \mid S / S \mid S >> S
\end{aligned}
$$

Semantic Spec

$f(1011_2) = 0011_2$



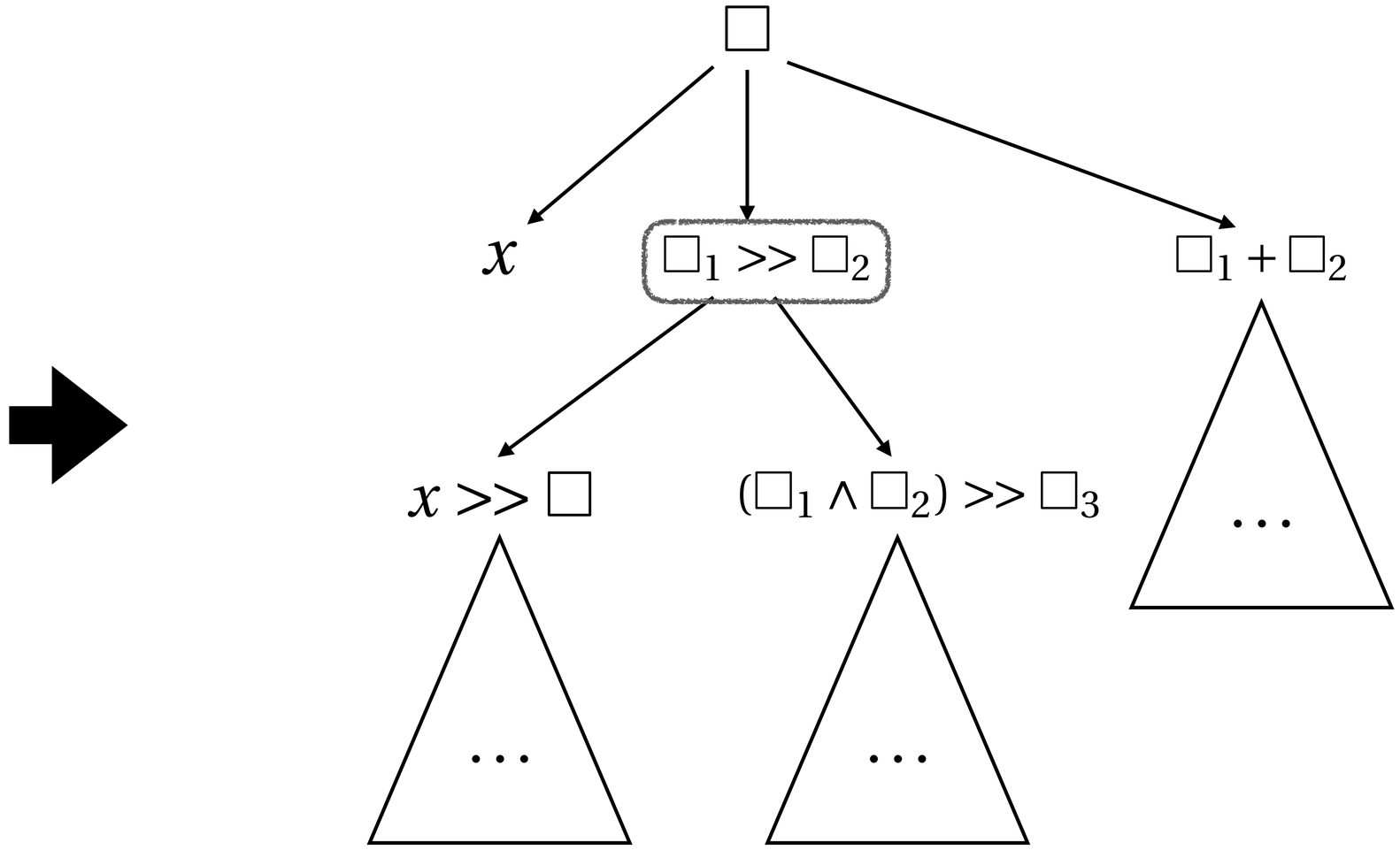
Pruning by Feasibility

Syntactic Spec

$$
\begin{aligned}
S \quad &\rightarrow \quad x \mid 0001_2 \\
&\mid \quad S \wedge S \mid S \vee S \mid S \oplus S \\
&\mid \quad S + S \mid S \times S \mid S / S \mid S >> S
\end{aligned}
$$

Semantic Spec

$f(1011_2) = 0011_2$



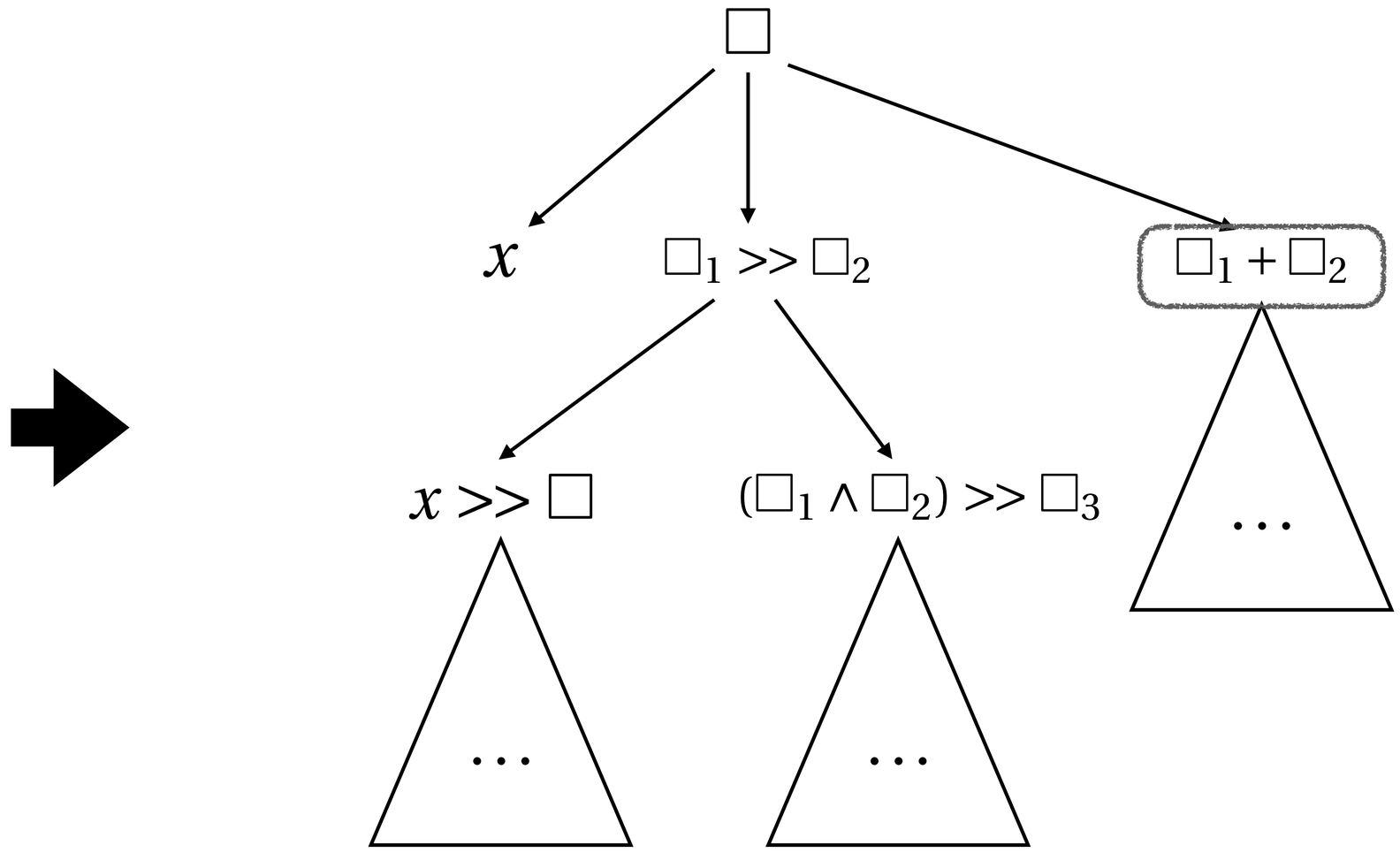
Pruning by Feasibility

Syntactic Spec

$$
\begin{aligned}
S \quad &\rightarrow \quad x \mid 0001_2 \\
&\mid \quad S \wedge S \mid S \vee S \mid S \oplus S \\
&\mid \quad S + S \mid S \times S \mid S / S \mid S >> S
\end{aligned}
$$

Semantic Spec

$f(1011_2) = 0011_2$



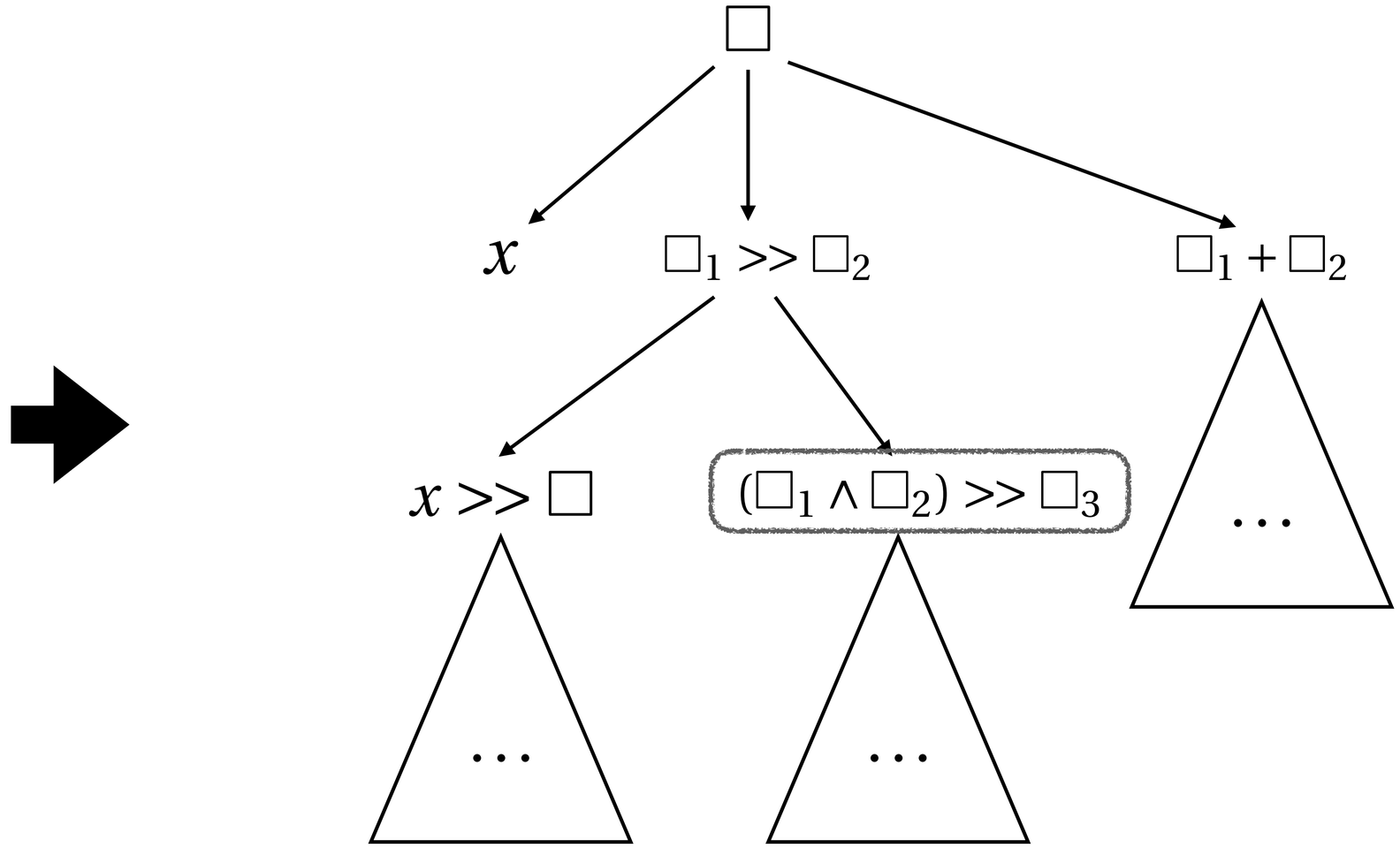
Pruning by Feasibility

Syntactic Spec

$$
\begin{aligned}
S \quad &\rightarrow \quad x \mid 0001_2 \\
&\mid \quad S \wedge S \mid S \vee S \mid S \oplus S \\
&\mid \quad S + S \mid S \times S \mid S / S \mid S >> S
\end{aligned}
$$

Semantic Spec

$f(1011_2) = 0011_2$



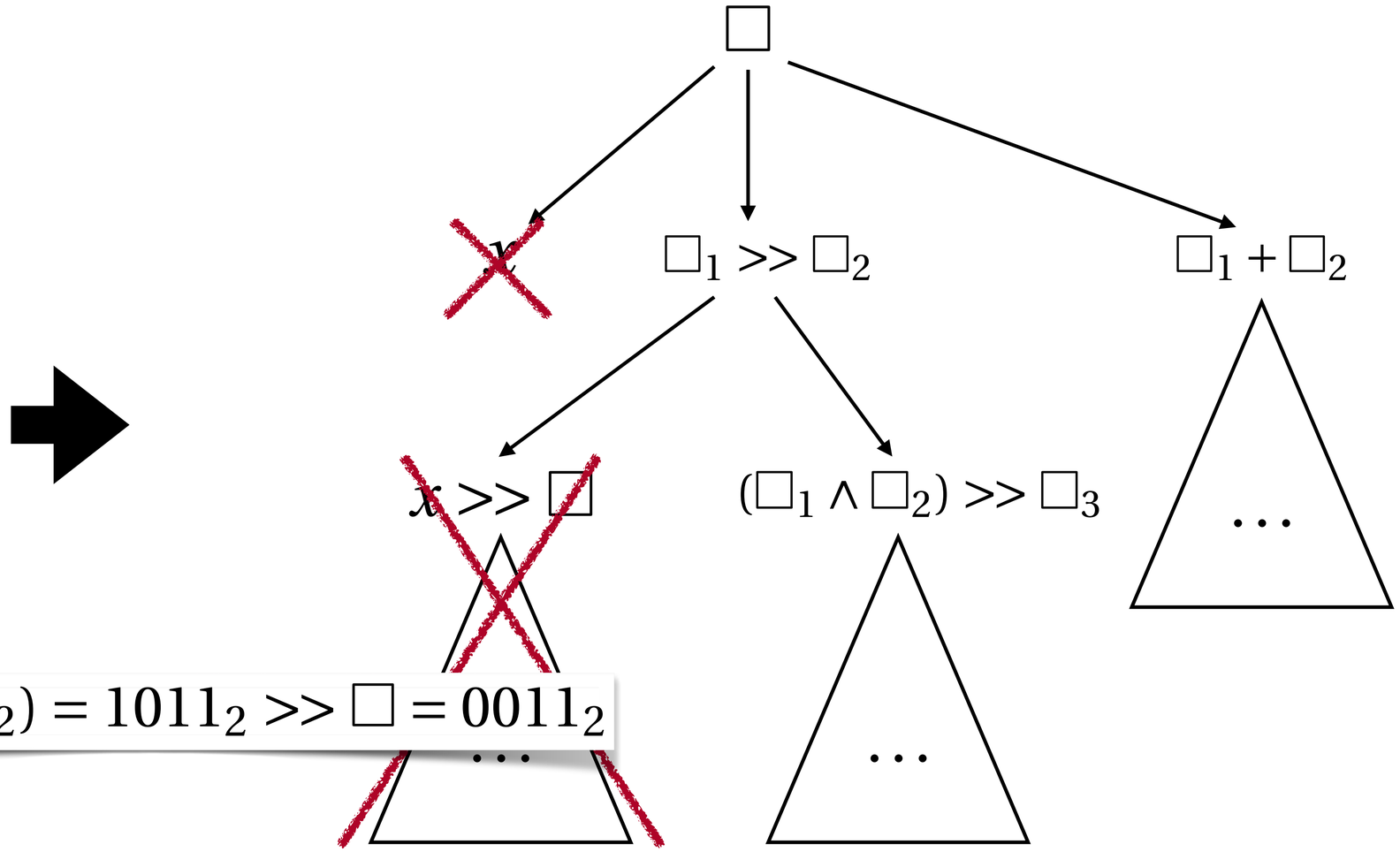
Pruning by Feasibility

Syntactic Spec

$$
\begin{array}{rcl}
S & \rightarrow & x \mid 0001_2 \\
 & \mid & S \wedge S \mid S \vee S \mid S \oplus S \\
 & \mid & S + S \mid S \times S \mid S / S \mid S >> S
\end{array}
$$

Semantic Spec

$f(1011_2) = 0011_2$



$$\nexists \square . f(1011_2) = 1011_2 >> \square = 0011_2$$

Pruning by Feasibility

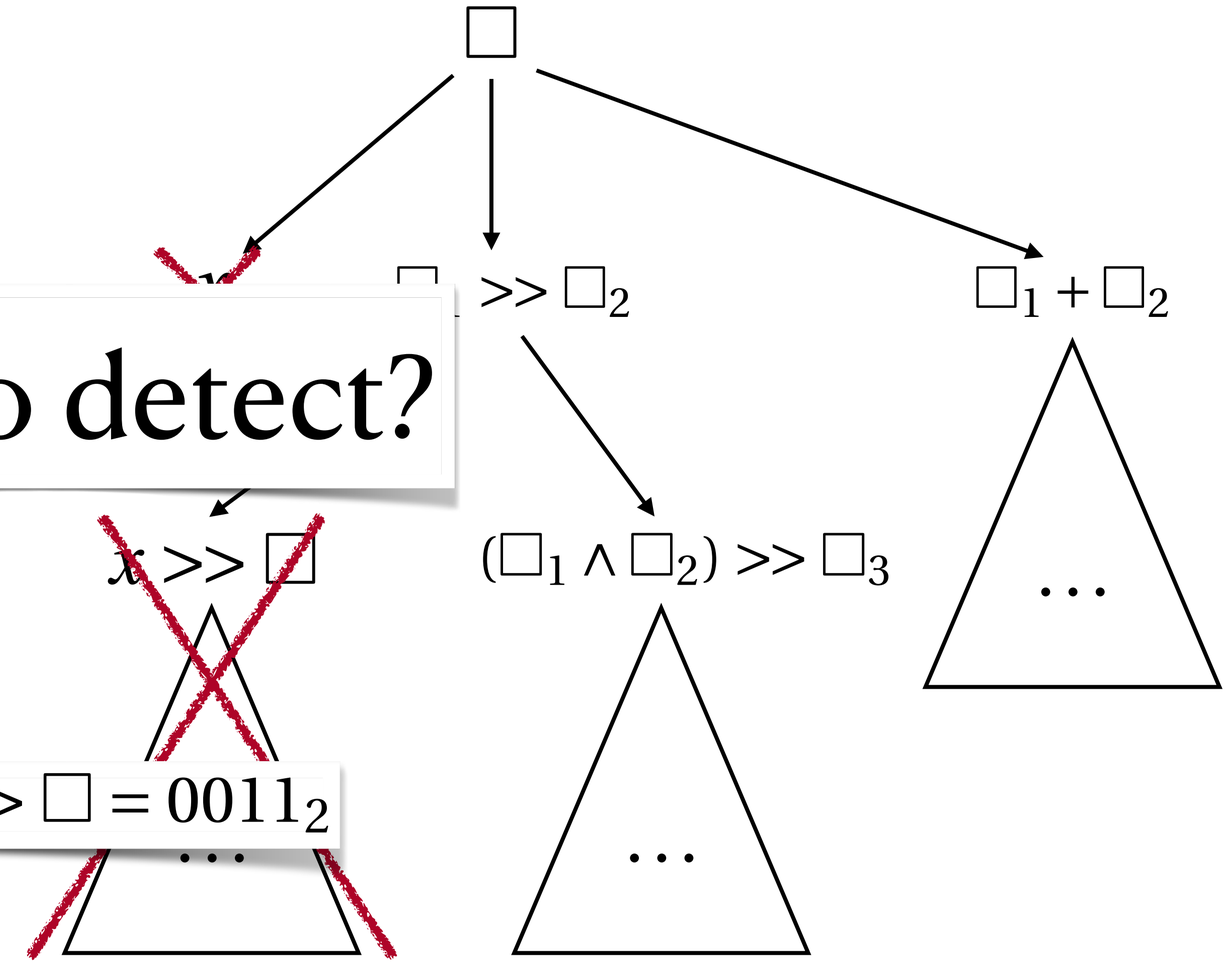
Syntactic Spec

$$
\begin{array}{rcl}
S & \rightarrow & x \mid 0001_2 \\
 & \mid & S \wedge S \mid S \vee S \mid S \oplus S \\
 & \mid & S + S \mid S \times S \mid S / S \mid S >> S
\end{array}
$$

Semantic Spec

$f(1011_2) = 0011_2$

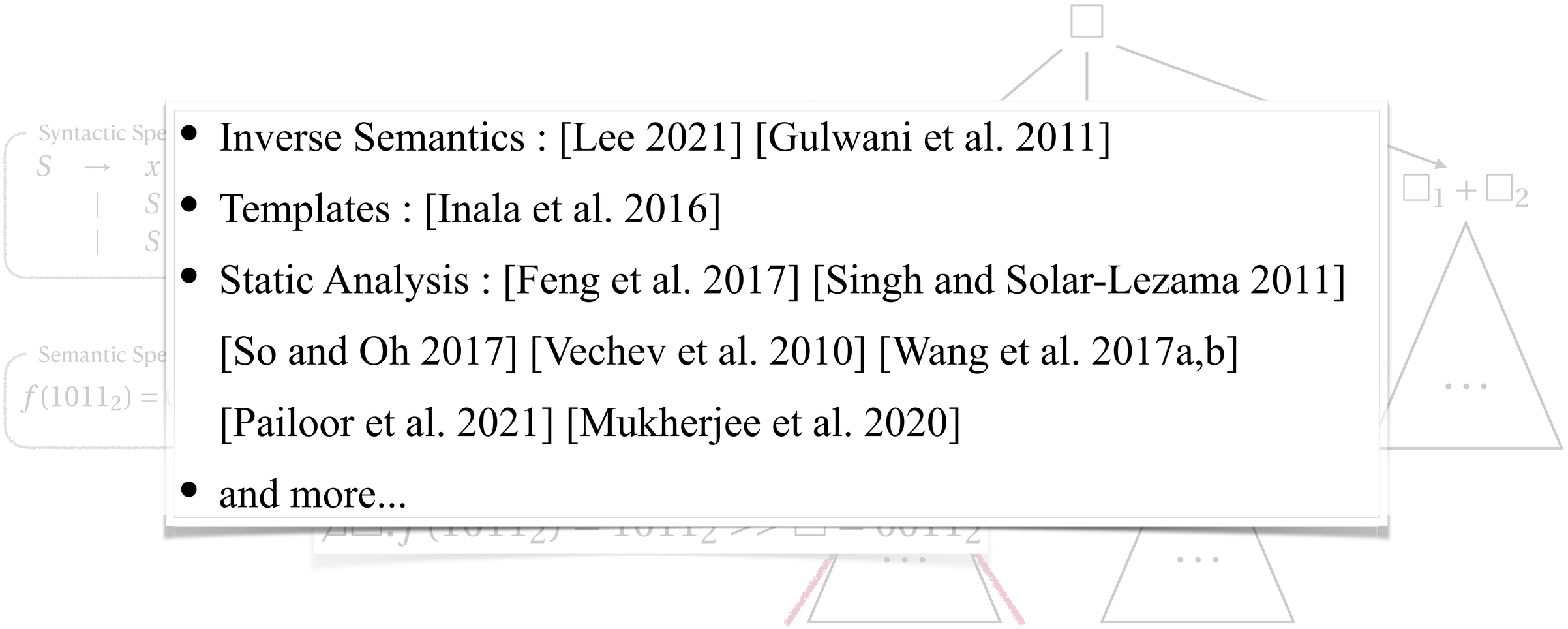
How to detect?



$\nexists \square . f(1011_2) = 1011_2 >> \square = 0011_2$

Pruning by Feasibility

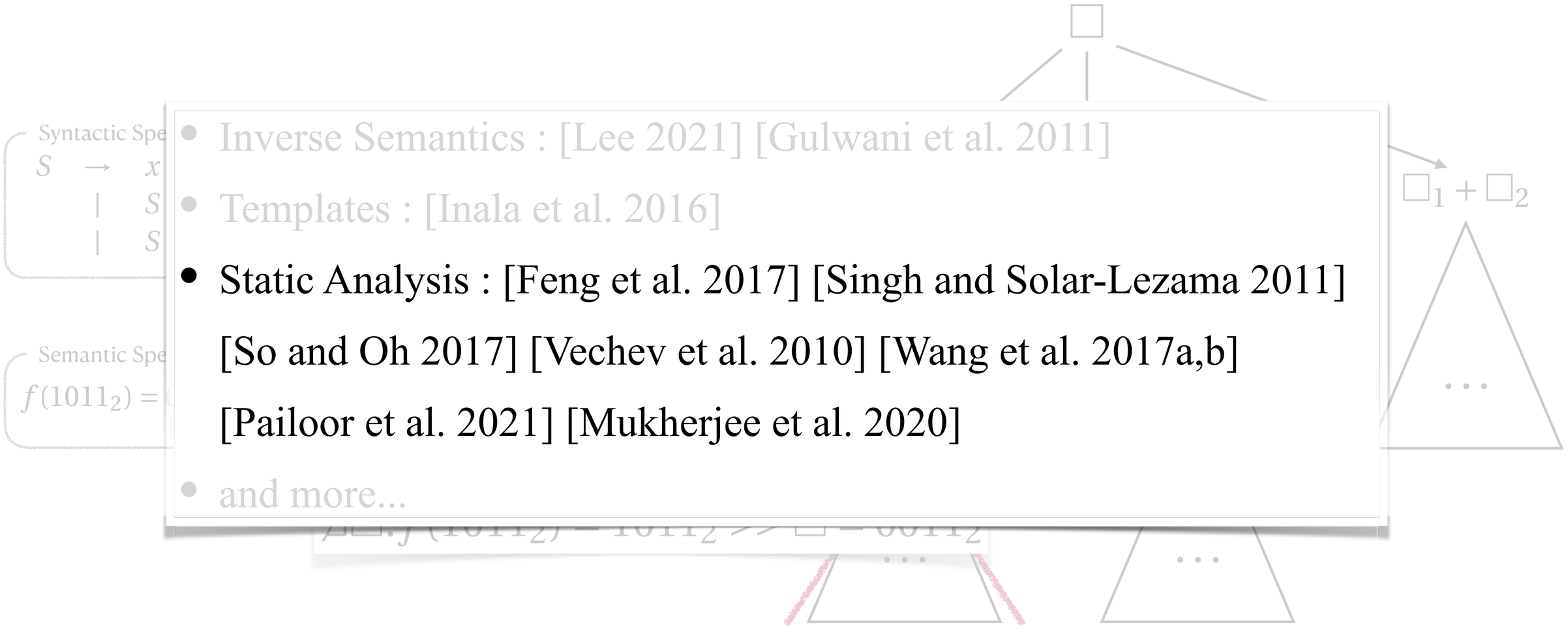
Existing Approaches



- Inverse Semantics : [Lee 2021] [Gulwani et al. 2011]
- Templates : [Inala et al. 2016]
- Static Analysis : [Feng et al. 2017] [Singh and Solar-Lezama 2011] [So and Oh 2017] [Vechev et al. 2010] [Wang et al. 2017a,b] [Pailoor et al. 2021] [Mukherjee et al. 2020]
- and more...

Pruning by Feasibility

Existing Approaches

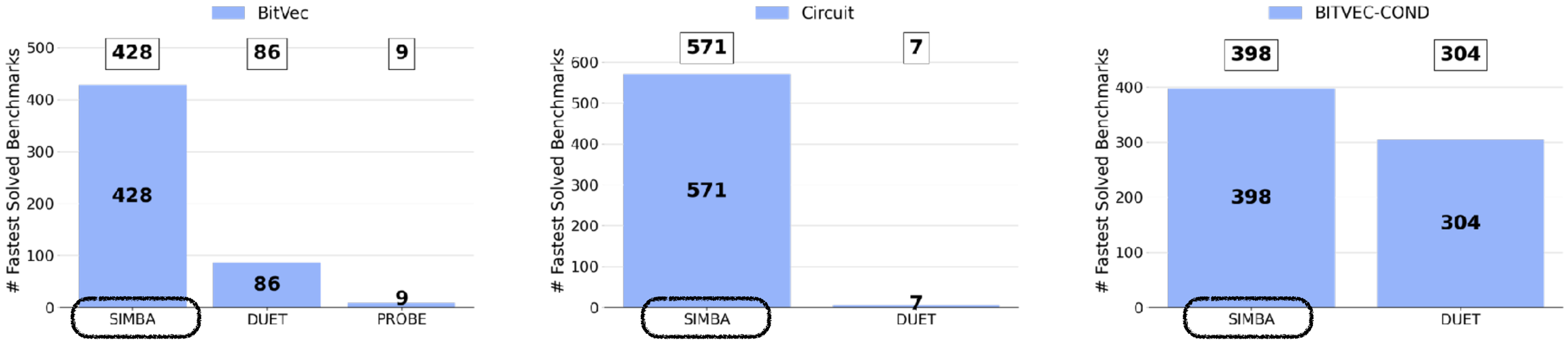


- Inverse Semantics : [Lee 2021] [Gulwani et al. 2011]
- Templates : [Inala et al. 2016]
- Static Analysis : [Feng et al. 2017] [Singh and Solar-Lezama 2011] [So and Oh 2017] [Vechev et al. 2010] [Wang et al. 2017a,b] [Pailoor et al. 2021] [Mukherjee et al. 2020]
- and more...

Our Performance

Comparing to 2 existing tools on 3 benchmark domains

- Our tool SIMBA using [Forward-Backward Static Analysis]
  - Outperformed state-of-the-art tools for 2 branch-free benchmarks
  - Comparable to state-of-the-art tools for a branch benchmark



of the fastest-solved problems for each domain

Using Forward Analysis

Checking only output feasibility

Semantic Spec

$f(1011_2) = 0011_2$

Candidate Partial Program

$f(x) = x \vee \square$

Forward

$x \mapsto 1011$

$\square \mapsto \top\top\top\top$

Using Forward Analysis

Checking only output feasibility

Semantic Spec

$f(1011_2) = 0011_2$

Candidate Partial Program

$f(x) = x \vee \square$

Forward

$$
\begin{array}{rcl}
x & \mapsto & 1011 \\
\square & \mapsto & \top\top\top\top \\
x \vee \square & \mapsto & 1\top 11 \not\sqsupseteq 0011
\end{array}
$$

Infeasible output

Limitation of Forward Analysis

Checking only output feasibility

Semantic Spec

$f(1011_2) = 0011_2$

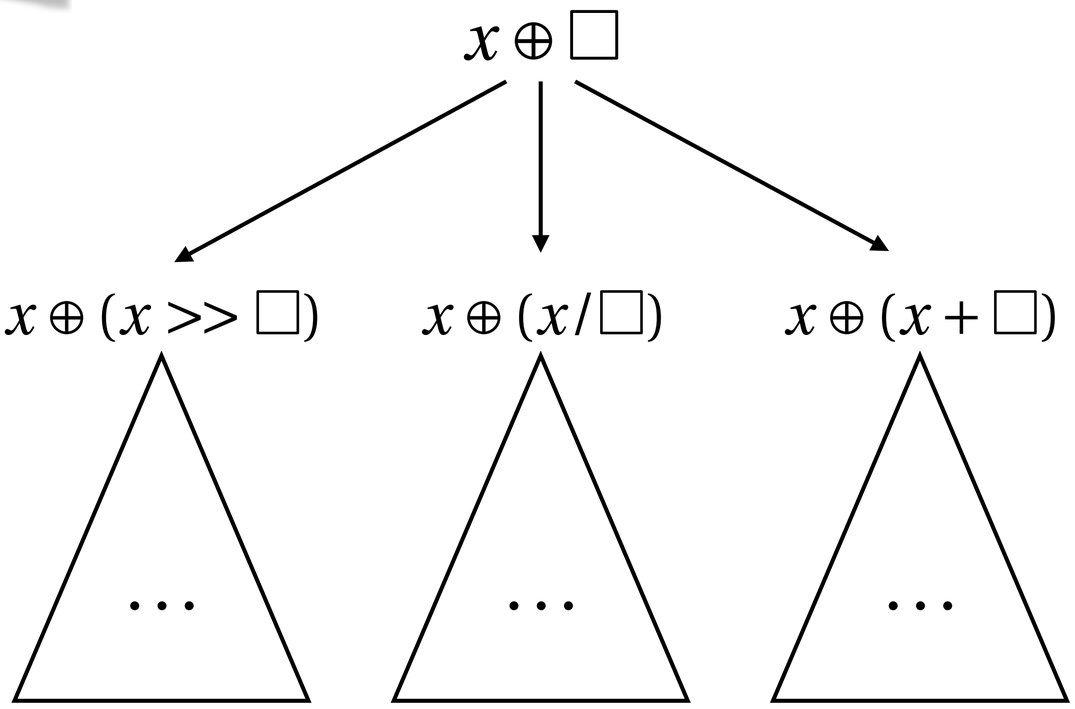
Candidate Partial Program

$f(x) = x \oplus \square$

Forward

$$
\begin{array}{rcl}
x & \mapsto & 1011 \\
\square & \mapsto & \top\top\top\top \\
x \oplus \square & \mapsto & \top\top\top\top \sqsupseteq 0011
\end{array}
$$

Nothing to do



Need: Backward Analysis Too

Output feasibility + Hole Precondition

Semantic Spec

$f(1011_2) = 0011_2$

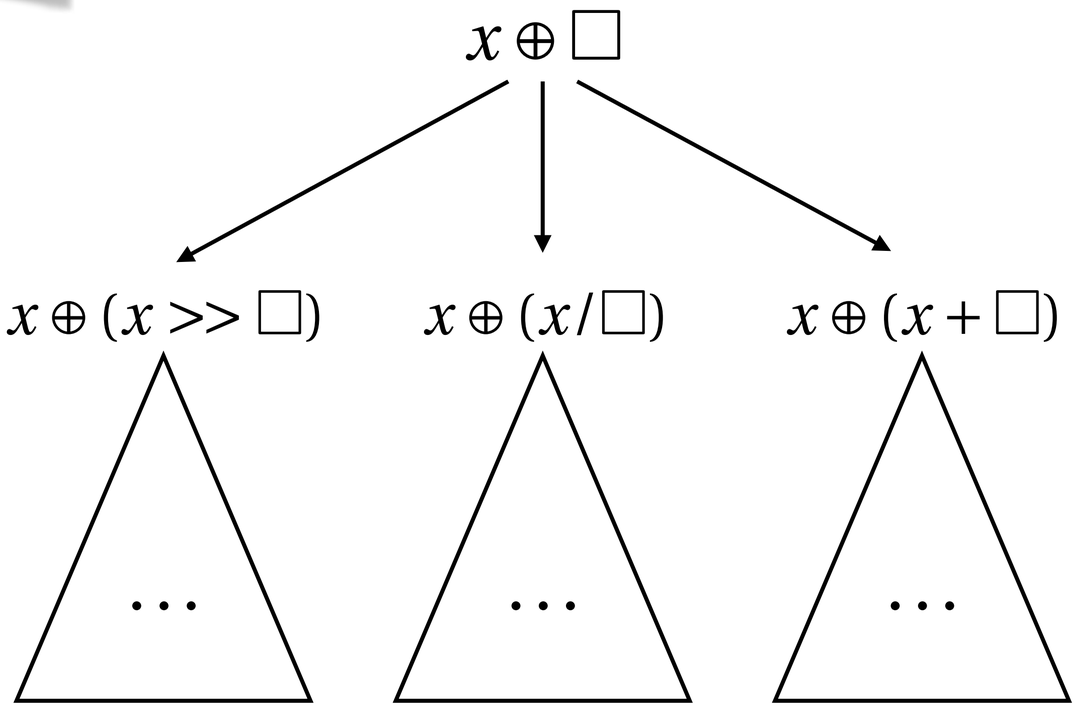
Candidate Partial Program

$f(x) = x \oplus \square$

Forward

$$x \mapsto 1011$$
$$\square \mapsto \top\top\top\top$$
$$x \oplus \square \mapsto 0011$$

Backward



Need: Backward Analysis Too

Output feasibility + Hole Precondition

Semantic Spec

$f(1011_2) = 0011_2$

Candidate Partial Program

$f(x) = x \oplus \square$

Forward

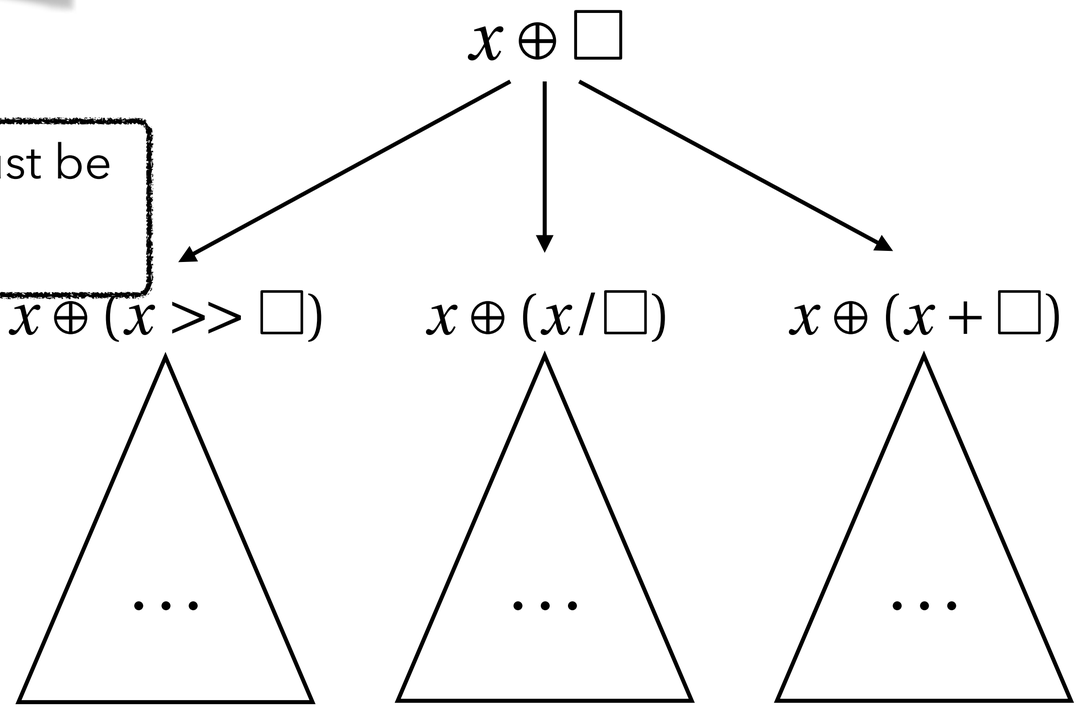
$x \mapsto 1011$

$\square \mapsto 1000$

$x \oplus \square \mapsto 0011$

value of hole must be 1000_2

Backward



Need: Backward Analysis Too

Output feasibility + Hole Precondition

Semantic Spec

$f(1011_2) = 0011_2$

Candidate Partial Program

$f(x) = x \oplus \square$

Forward

$x \mapsto 10$

$\square \mapsto 10$

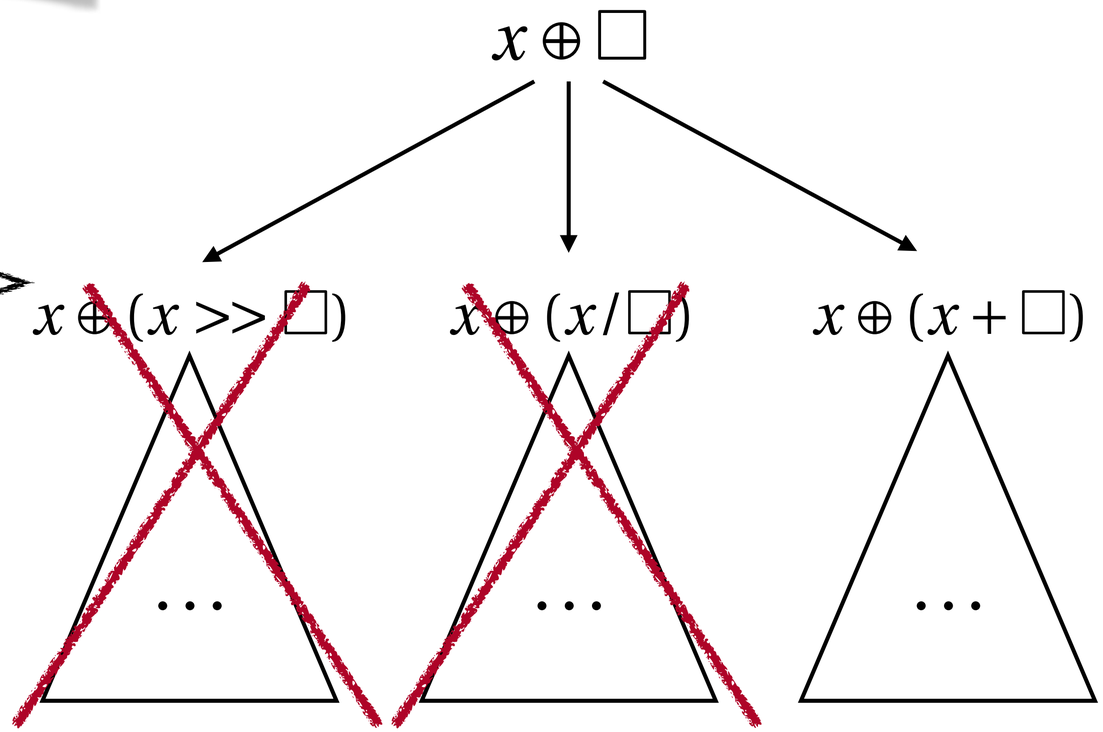
$x \oplus \square \mapsto 00$

prune infeasible parts

$x >> \square \neq 1000_2$

$x / \square$

Backward



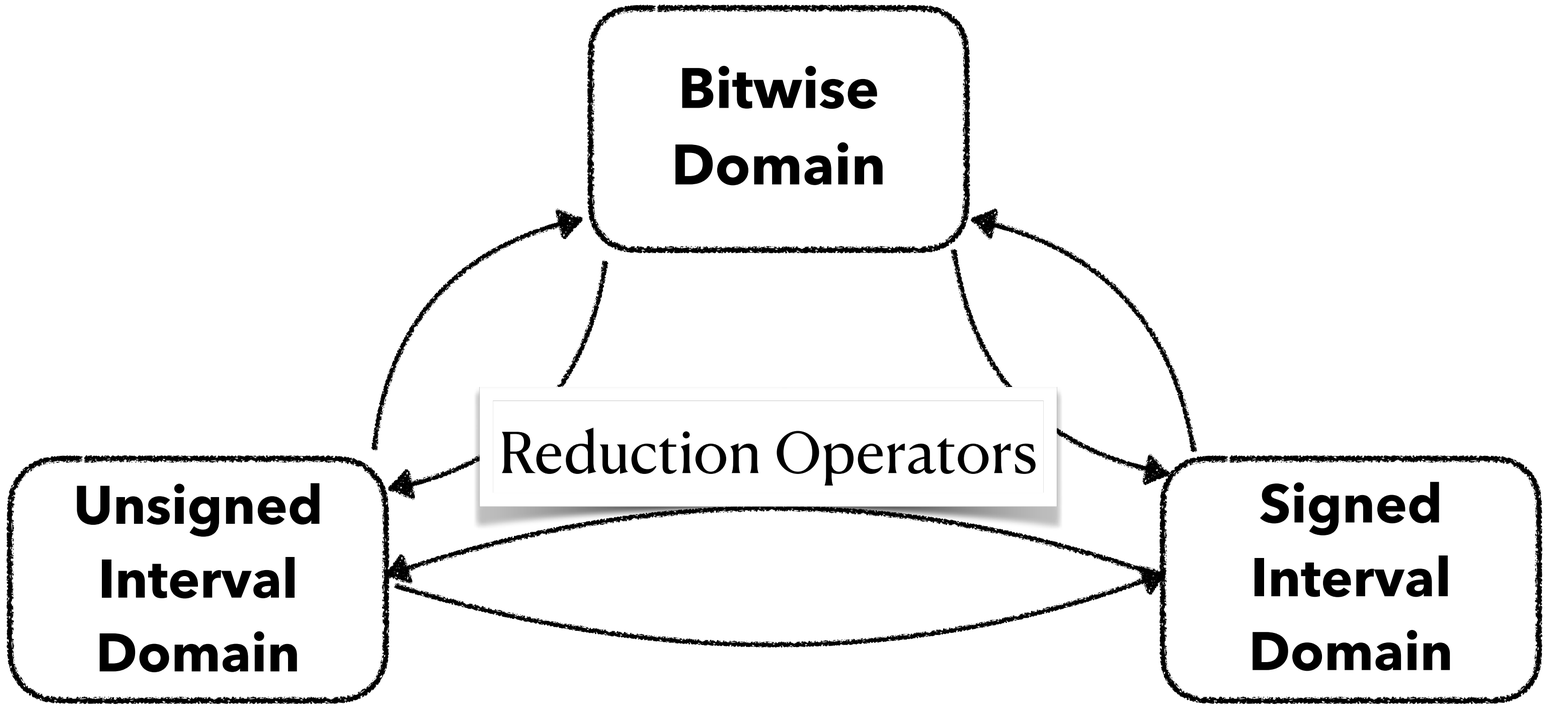
Challenges

Challenge 1: Precision & Cost Balance

Challenge 2: Precise Backward Analysis

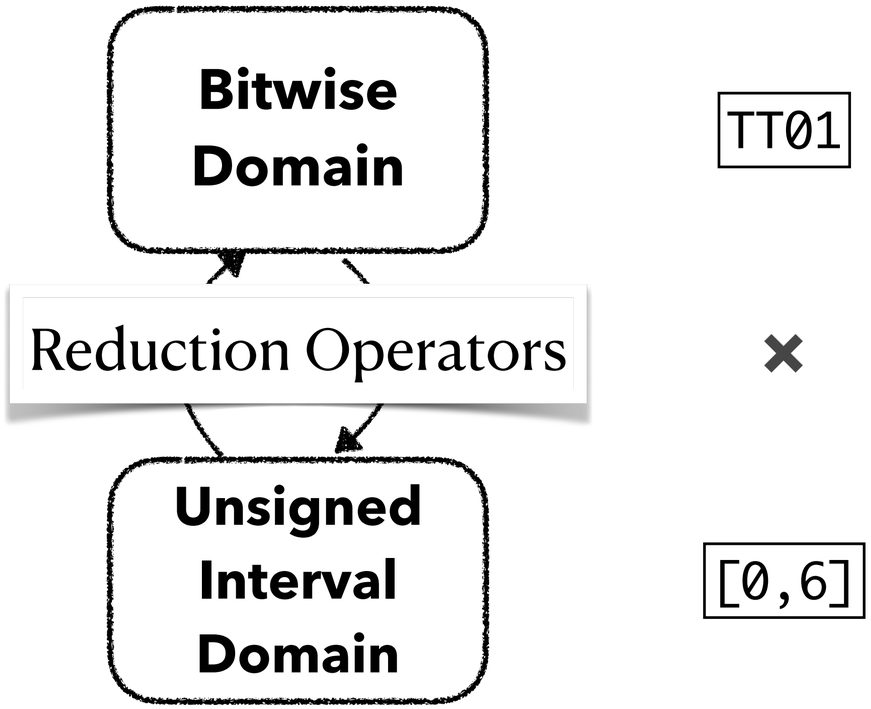
Challenge 1: Precision & Cost Balance

- Reduced Product Domain of well-known and simple domains
- No more complex domain (e.g., relational domain) was necessary



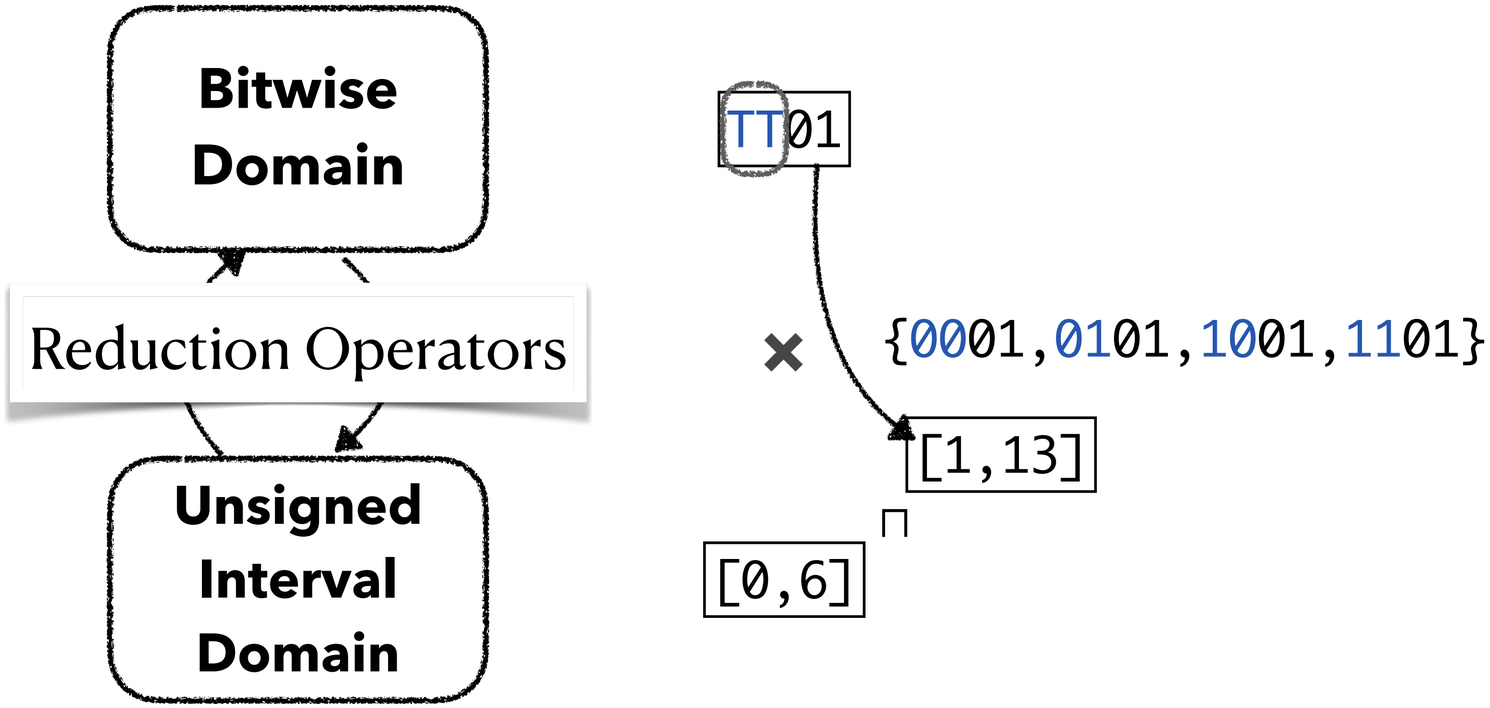
Challenge 1: Precision & Cost Balance

- Reduction Example: Bitwise ✖ Unsigned Interval



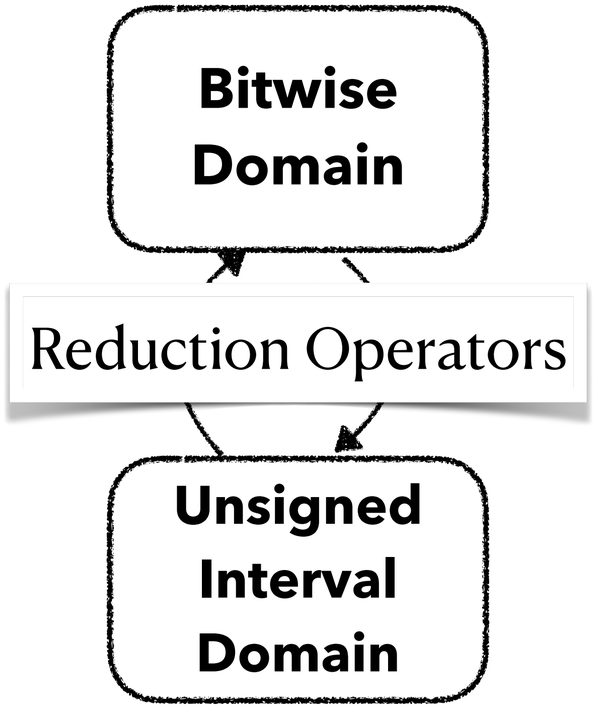
Challenge 1: Precision & Cost Balance

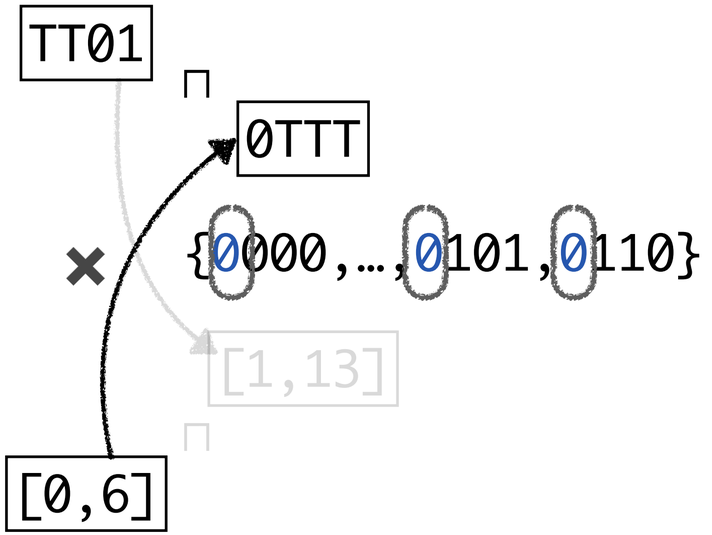
- Reduction Example: Bitwise ✖ Unsigned Interval



Challenge 1: Precision & Cost Balance

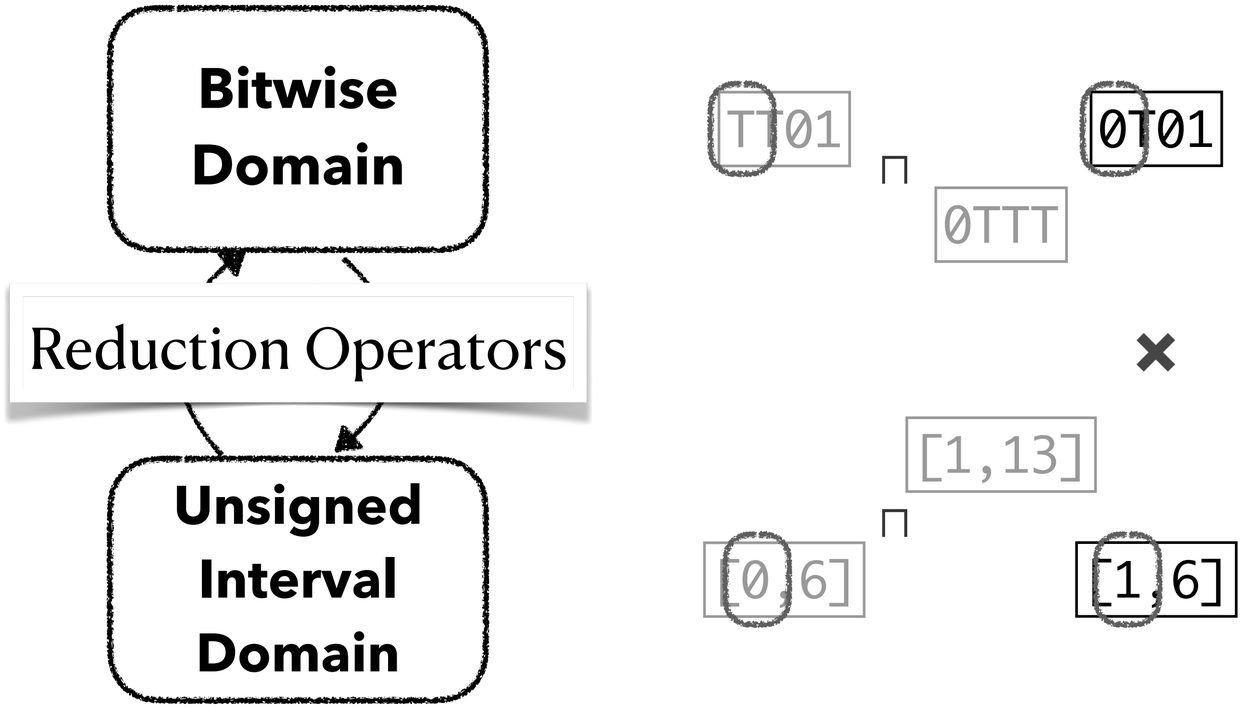
- Reduction Example: Bitwise ✖ Unsigned Interval





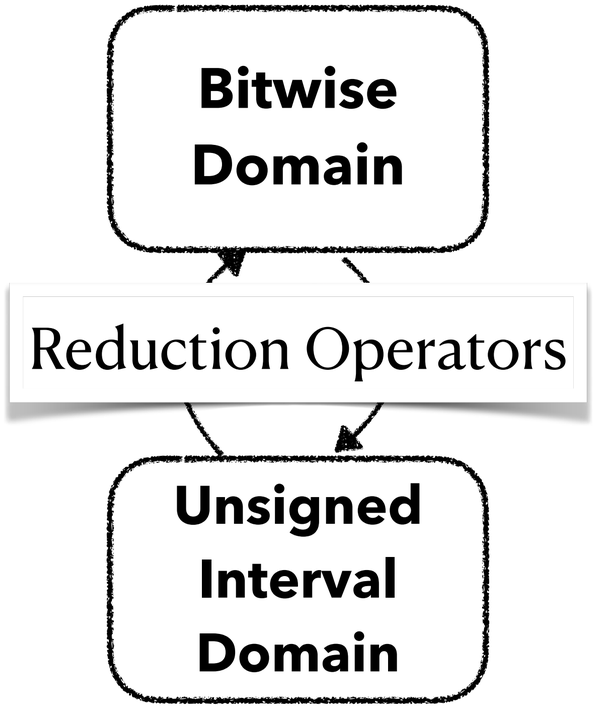
Challenge 1: Precision & Cost Balance

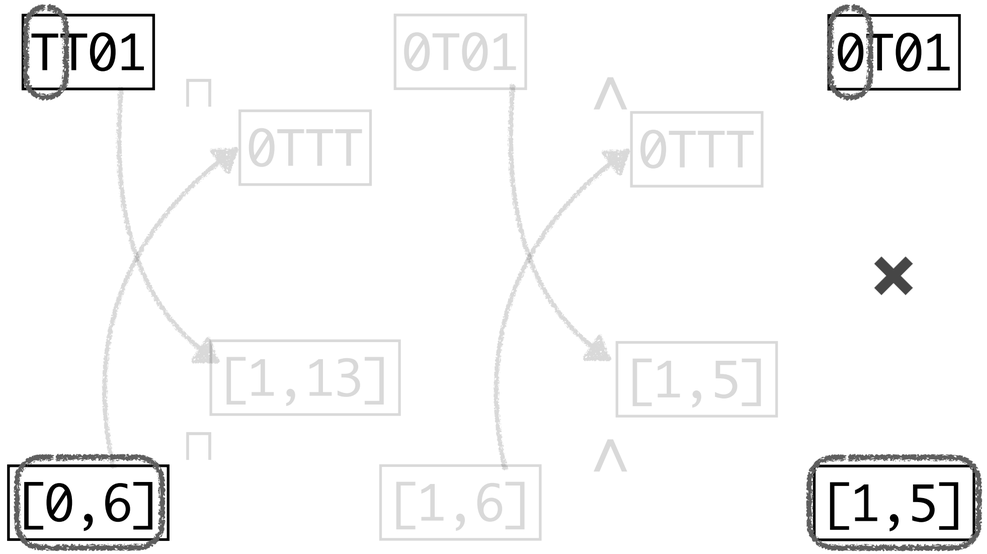
- Reduction Example: Bitwise ✖ Unsigned Interval



Challenge 1: Precision & Cost Balance

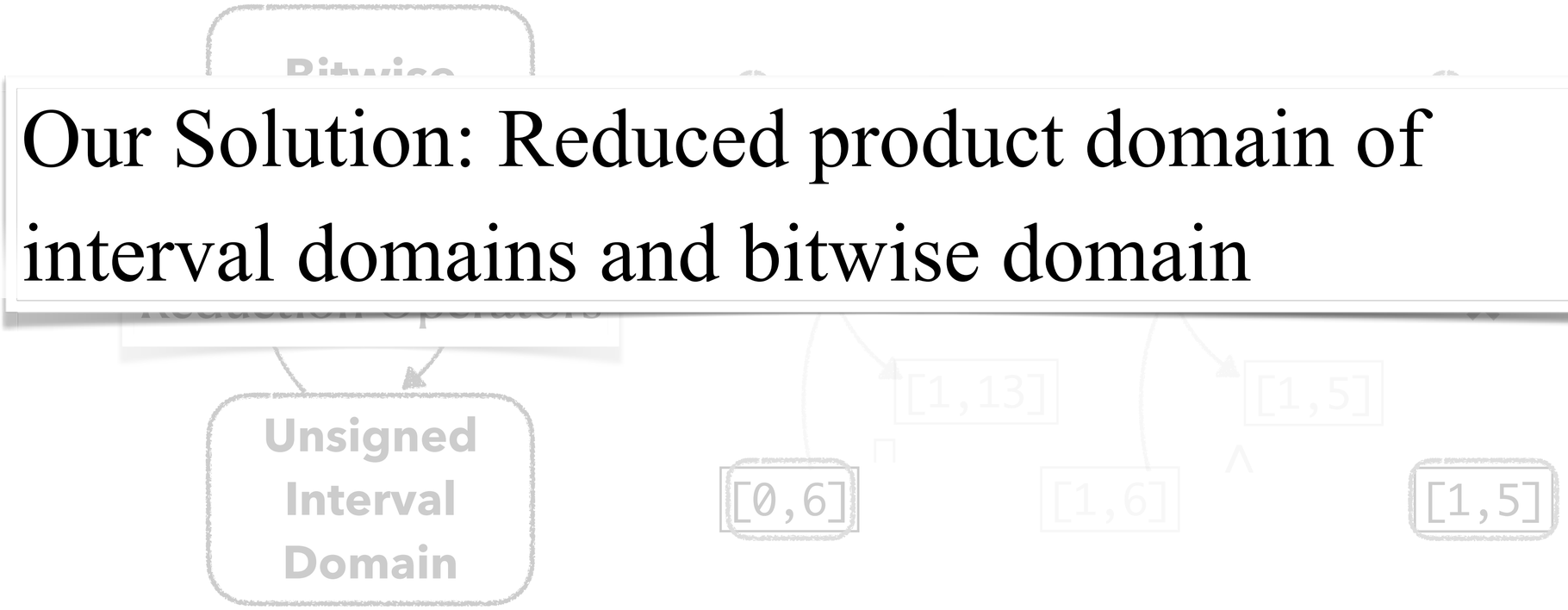
- Reduction Example: Bitwise ✖ Unsigned Interval





Challenge 1: Precision & Cost Balance

- Reduction Example: Bitwise ✖ Unsigned Interval



Our Solution: Reduced product domain of interval domains and bitwise domain

Challenge 2: Precise Backward Analysis by Selective Finite Concretization

Candidate Partial Program

$f(x) = x \wedge (x \times \square)$

Semantic Spec

$f(1011_2) = 0011_2$

Forward

$$
\begin{array}{rcll}
x & \mapsto & 1011 & \\
\square & \mapsto & \top\top\top\top & \\
x \times \square & \mapsto & \top\top\top\top & \\
x \wedge (x \times \square) & \mapsto & \top\top\top\top & \sqsupseteq 0011
\end{array}
$$

Challenge 2: Precise Backward Analysis by Selective Finite Concretization

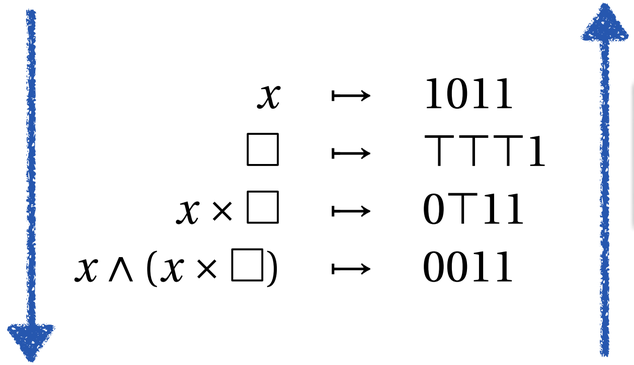
Candidate Partial Program

$f(x) = x \wedge (x \times \square)$

Semantic Spec

$f(1011_2) = 0011_2$

Forward



$$
\begin{array}{rcl}
x & \mapsto & 1011 \\
\square & \mapsto & \top\top\top 1 \\
x \times \square & \mapsto & 0\top 11 \\
x \wedge (x \times \square) & \mapsto & 0011
\end{array}
$$

Can we do better?

Backward

Challenge 2: Precise Backward Analysis by Selective Finite Concretization

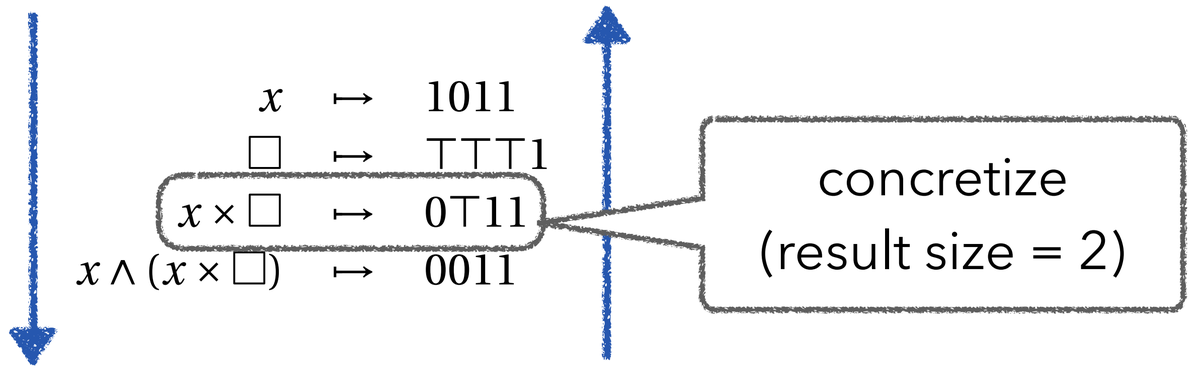
Candidate Partial Program

$f(x) = x \wedge (x \times \square)$

Semantic Spec

$f(1011_2) = 0011_2$

Forward



Backward

Challenge 2: Precise Backward Analysis by Selective Finite Concretization

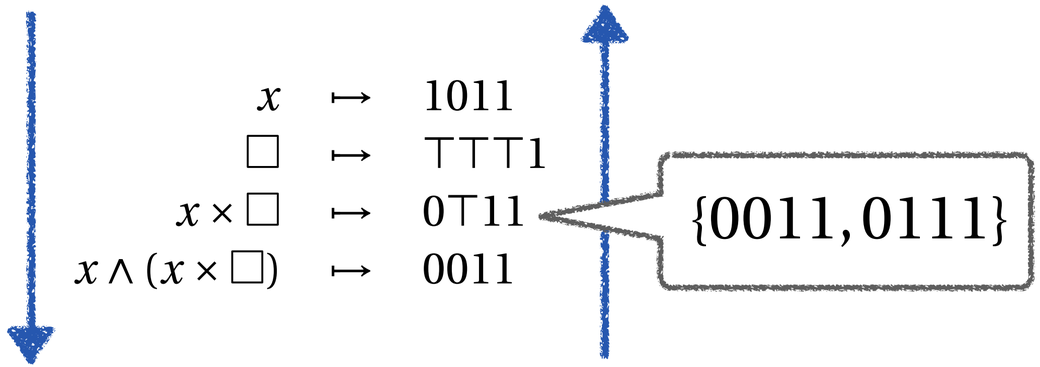
Candidate Partial Program

$f(x) = x \wedge (x \times \square)$

Semantic Spec

$f(1011_2) = 0011_2$

Forward



Backward

Challenge 2: Precise Backward Analysis by Selective Finite Concretization

Candidate Partial Program

$f(x) = x \wedge (x \times \square)$

Semantic Spec

$f(1011_2) = 0011_2$

Forward

$$
\begin{array}{rcl}
x & \mapsto & 1011 \\
\square & \mapsto & \top\top\top 1 \\
x \times \square & \mapsto & 0\top 11 \\
x \wedge (x \times \square) & \mapsto & 0011
\end{array}
$$

we get 2 equations

$$
\begin{array}{lcl}
1011 \times \square_0 & = & 0011 \\
1011 \times \square_1 & = & 0111
\end{array}
$$

Backward

Challenge 2: Precise Backward Analysis by Selective Finite Concretization

Candidate Partial Program

$f(x) = x \wedge (x \times \square)$

Semantic Spec

$f(1011_2) = 0011_2$

Forward

$$
\begin{array}{rcl}
x & \mapsto & 1011 \\
\square & \mapsto & \top\top\top 1 \\
x \times \square & \mapsto & 0\top 11 \\
x \wedge (x \times \square) & \mapsto & 0011
\end{array}
$$

Backward

we get 2 equations

$$
\begin{array}{lcl}
1011 \times \square_0 & = & 0011 \mod 16 \\
1011 \times \square_1 & = & 0111 \mod 16
\end{array}
$$

Incorrect approach: transposition

$$
\begin{array}{lcl}
1011 \times \square_0 & = & 0011/1011 \\
1011 \times \square_1 & = & 0111/1011
\end{array}
$$

Challenge 2: Precise Backward Analysis by Selective Finite Concretization

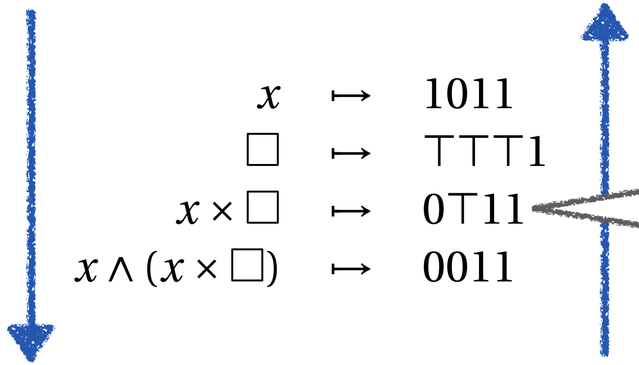
Candidate Partial Program

$f(x) = x \wedge (x \times \square)$

Semantic Spec

$f(1011_2) = 0011_2$

Forward



$$
\begin{array}{rcl}
x & \mapsto & 1011 \\
\square & \mapsto & \top\top\top 1 \\
x \times \square & \mapsto & 0\top 11 \\
x \wedge (x \times \square) & \mapsto & 0011
\end{array}
$$

Backward

we get 2 equations

$$
\begin{array}{lcl}
1011 \times \square_0 & = & 0011 \mod 16 \\
1011 \times \square_1 & = & 0111 \mod 16
\end{array}
$$

by extended Euclidean algorithm,

$$
\begin{array}{lcl}
\square_0 & = & 1001 \\
\square_1 & = & 0101
\end{array}
$$

Challenge 2: Precise Backward Analysis by Selective Finite Concretization

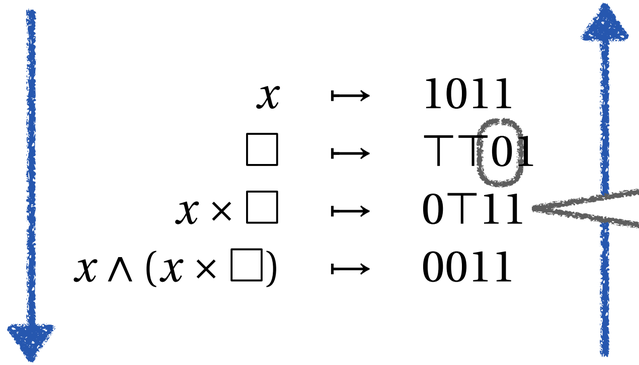
Candidate Partial Program

$f(x) = x \wedge (x \times \square)$

Semantic Spec

$f(1011_2) = 0011_2$

Forward



$$
\begin{array}{rcl}
x & \mapsto & 1011 \\
\square & \mapsto & \top\top 01 \\
x \times \square & \mapsto & 0\top 11 \\
x \wedge (x \times \square) & \mapsto & 0011
\end{array}
$$

Backward

we get 2 equations

$$
\begin{array}{lcll}
1011 \times \square_0 & = & 0011 & \bmod 16 \\
1011 \times \square_1 & = & 0111 & \bmod 16
\end{array}
$$

by extended Euclidean algorithm,

$$
\begin{array}{rcl}
\square_0 & = & 1001 \\
\square_1 & = & 0101 \\
\square & = & \square_0 \sqcup \square_1 = \top\top 01
\end{array}
$$

Challenge 2: Precise Backward Analysis by Selective Finite Concretization

Candidate Partial Program

$f(x) = x \wedge (x \times \square)$

Semantic Spec

$f(1011_2) = 0011_2$

Forward

Backward

Our Solution: Use concretization for precise analysis

$x \wedge (x \times \square) \quad \mapsto \quad 0011$

$1011 \times \square_0 \quad = \quad 0011 \mod 16$

$1011 \times \square_1 \quad = \quad 0111 \mod 16$

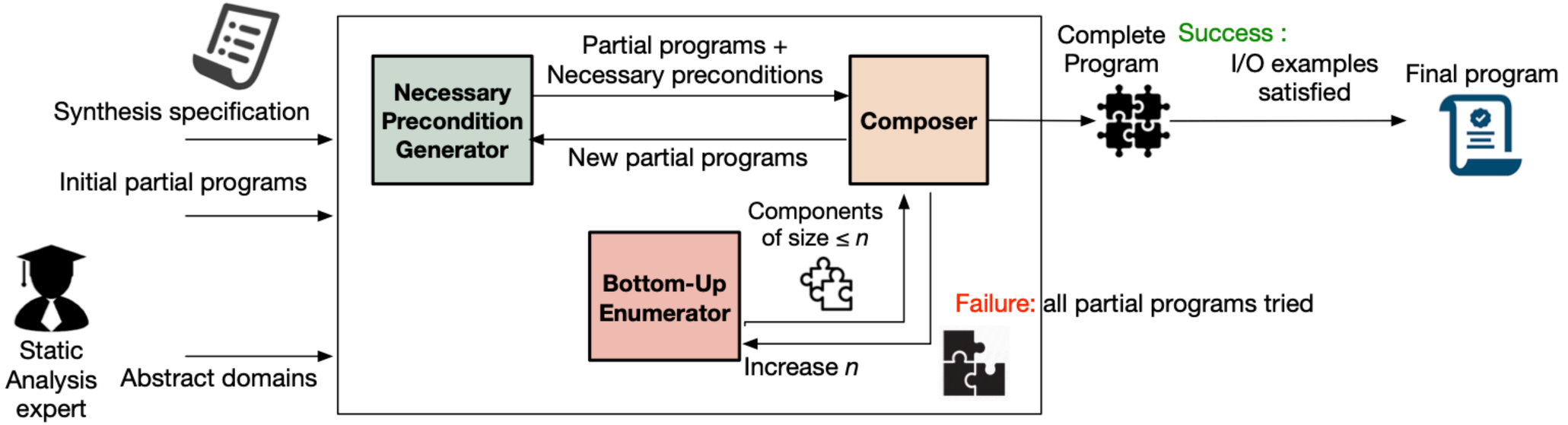
$\square_0 \quad = \quad 1001$

$\square_1 \quad = \quad 0101$

$\square \quad = \quad \square_0 \sqcup \square_1 = \top\top 01$

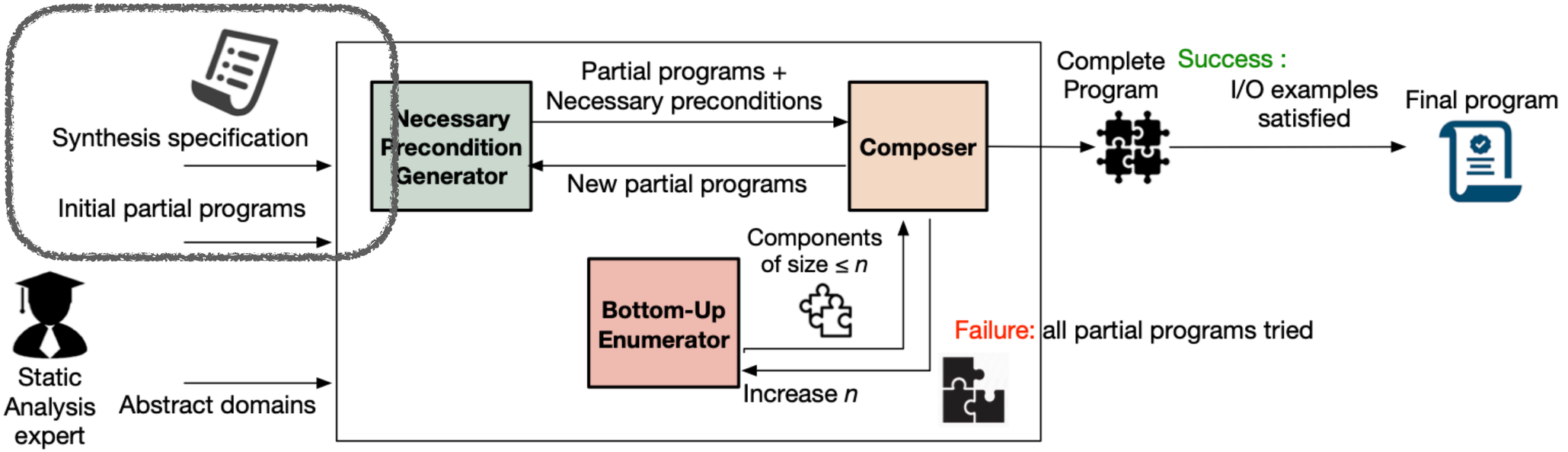
Realized: SIMBA

- **S**ynthesis from **I**nductive specification e**M**powered by **B**idirectional **A**bstract interpretation
- Available at https://github.com/yhyoon/simba



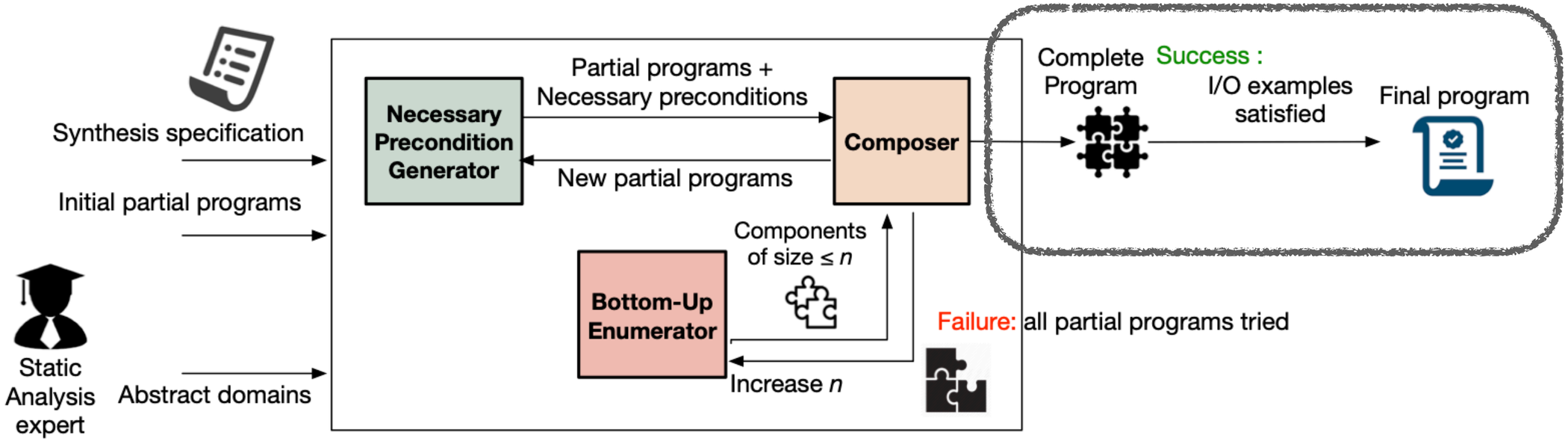
Realized: SIMBA

- **S**ynthesis from **I**nductive specification e**M**powered by **B**idirectional **A**bstract interpretation
- Available at https://github.com/yhyoon/simba

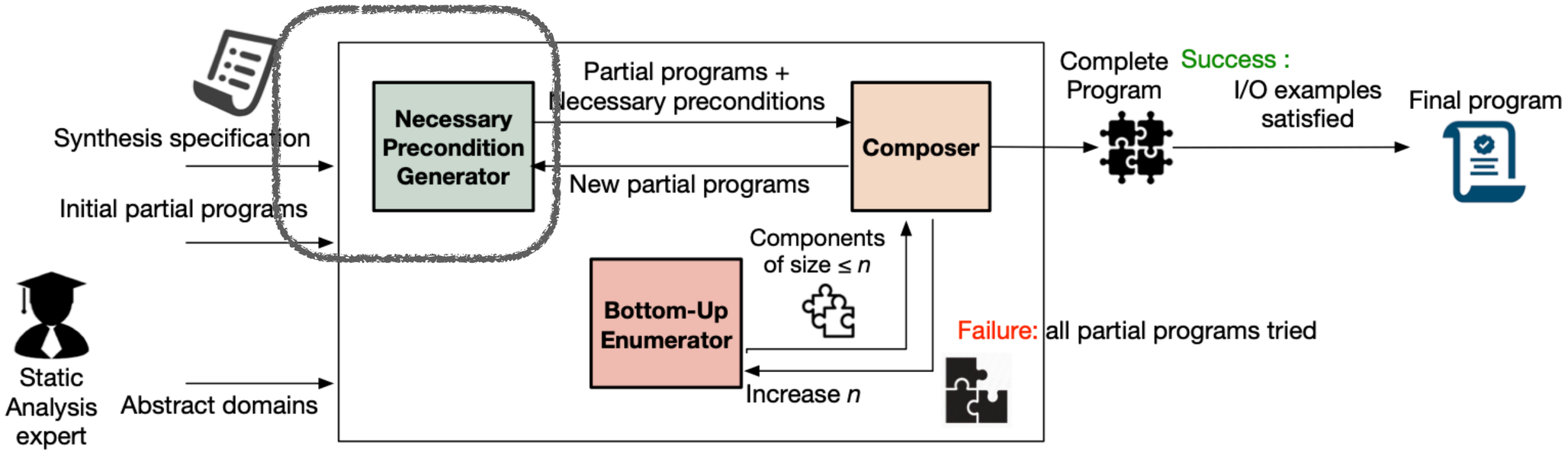


Realized: SIMBA

- **S**ynthesis from **I**nductive specification e**M**powered by **B**idirectional **A**bstract interpretation
- Available at https://github.com/yhyoon/simba



Synthesis Algorithm



Synthesis Algorithm

Example Initial Partial Programs

$(\square \oplus x) >> 0001_2$

$(\square / x) >> 0001_2$

Initial partial programs
chosen for illustration purpose

Spec

$$
\begin{array}{lll}
S & \rightarrow & x \mid 0001_2 \\
 & \mid & S \wedge S \mid S \vee S \mid S \oplus S \\
 & \mid & S + S \mid S \times S \mid S / S \mid S >> S
\end{array}
$$

$f(1011_2) = 0011_2$

Goal

$f(x) = ((x + 0001_2) \oplus x) >> 0001_2$

Synthesis Algorithm

Analyze Partial Programs (1/2)

$(\square \oplus x) >> 0001_2$

Forward Analysis

$\square$	$\mapsto$	$\top\top\top\top$
x	$\mapsto$	1011
$\square \oplus x$	$\mapsto$	$\top\top\top\top$
$(\square \oplus x) >> 0001_2$	$\mapsto$	$\top\top\top\top$

$(\square / x) >> 0001_2$

Spec

$$
\begin{array}{rcl}
S & \rightarrow & x \mid 0001_2 \\
 & \mid & S \wedge S \mid S \vee S \mid S \oplus S \\
 & \mid & S + S \mid S \times S \mid S / S \mid S >> S
\end{array}
$$

$f(1011_2) = 0011_2$

Goal

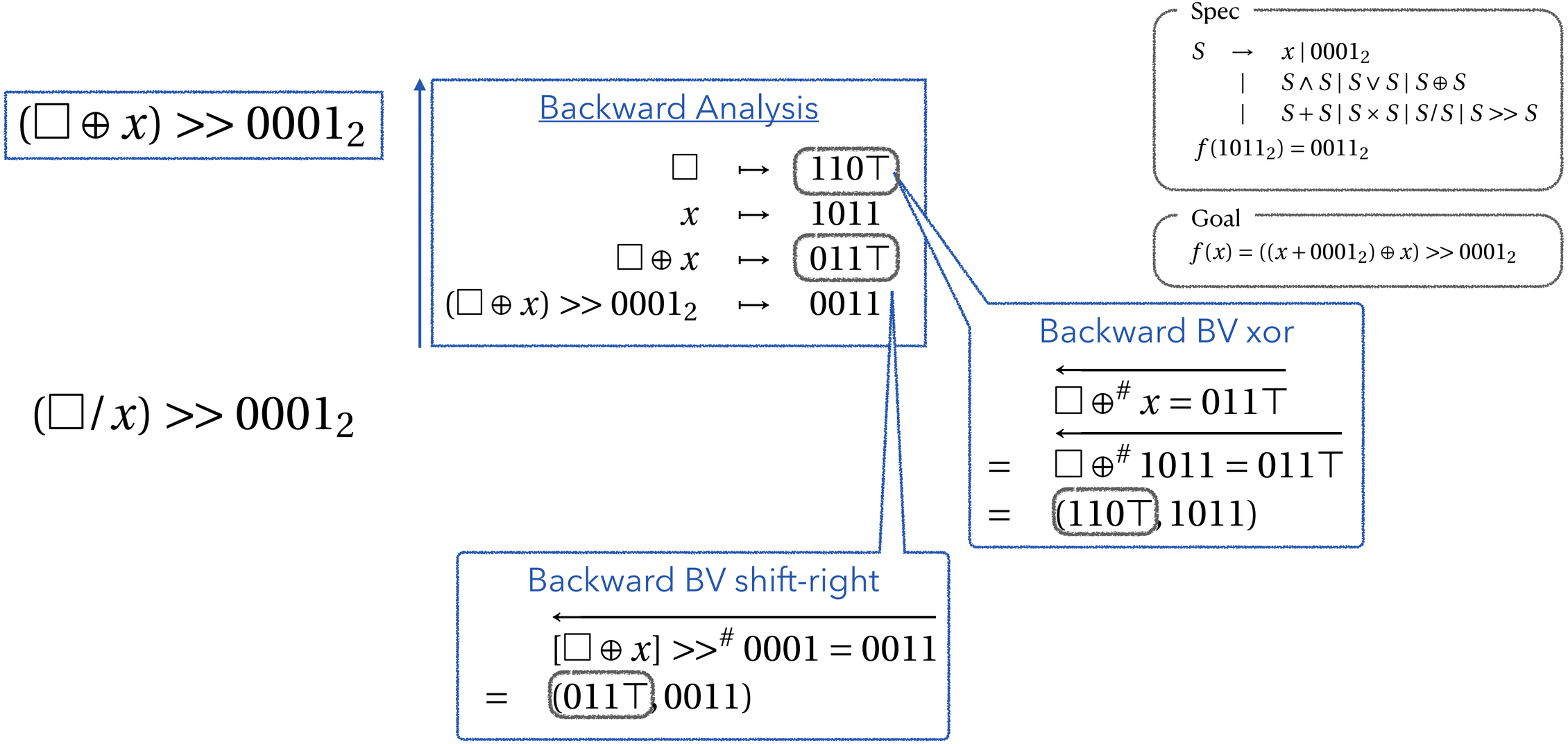
$f(x) = ((x + 0001_2) \oplus x) >> 0001_2$

Synthesis Algorithm

Analyze Partial Programs (1/2)

$(\square \oplus x) >> 0001_2$

$(\square / x) >> 0001_2$



Synthesis Algorithm

Analyze Partial Programs (2/2)

$$(\square \oplus x) >> 0001_2 \rightarrow$$

$$\begin{array}{rcl} \square & \mapsto & 110\top \\ x & \mapsto & 1011 \\ \square \oplus x & \mapsto & 011\top \\ (\square \oplus x) >> 0001_2 & \mapsto & 0011 \end{array}$$

Spec

$$\begin{array}{rcl} S & \rightarrow & x \mid 0001_2 \\ & \mid & S \wedge S \mid S \vee S \mid S \oplus S \\ & \mid & S + S \mid S \times S \mid S / S \mid S >> S \end{array}$$

$$f(1011_2) = 0011_2$$

Goal

$$f(x) = ((x + 0001_2) \oplus x) >> 0001_2$$

$$(\square / x) >> 0001_2$$

Forward Analysis

$$\begin{array}{rcl} \square & \mapsto & \top\top\top\top \\ x & \mapsto & 1011 \\ \square / x & \mapsto & 000\top \end{array}$$

$$\begin{array}{rcl} \square & = & [0, 15] \\ x & = & [11, 11] \\ \xrightarrow{\quad} & & \\ \square /^{\#} x & = & [l_S \,/\, h_x,\ h_S \,/\, l_x] \\ & = & [0 \,/\, 11,\ 15 \,/\, 11] \\ & = & [0, 1] \end{array}$$

Synthesis Algorithm

Analyze Partial Programs (2/2)

$(\square \oplus x) >> 0001_2 \rightarrow$

$\square$	$\mapsto$	$110\top$
x	$\mapsto$	1011
$\square \oplus x$	$\mapsto$	$011\top$
$(\square \oplus x) >> 0001_2$	$\mapsto$	0011

Spec

$$
\begin{array}{rcl}
S & \rightarrow & x \mid 0001_2 \\
 & \mid & S \wedge S \mid S \vee S \mid S \oplus S \\
 & \mid & S + S \mid S \times S \mid S / S \mid S >> S
\end{array}
$$

$f(1011_2) = 0011_2$

Goal

$f(x) = ((x + 0001_2) \oplus x) >> 0001_2$

$(\square / x) >> 0001_2$

Forward Analysis

$\square$	$\mapsto$	$\top\top\top\top$
x	$\mapsto$	1011
$\square / x$	$\mapsto$	$000\top$
$(\square / x) >> 0001_2$	$\mapsto$	$0000 \not\equiv 0011$

Synthesis Algorithm

Analyze Partial Programs (2/2)

$(\square \oplus x) >> 0001_2 \rightarrow$

$\square$	$\mapsto$	$110\top$
x	$\mapsto$	1011
$\square \oplus x$	$\mapsto$	$011\top$
$(\square \oplus x) >> 0001_2$	$\mapsto$	0011

Spec

$$
\begin{array}{rcl}
S & \rightarrow & x \mid 0001_2 \\
 & \mid & S \wedge S \mid S \vee S \mid S \oplus S \\
 & \mid & S + S \mid S \times S \mid S / S \mid S >> S
\end{array}
$$

$f(1011_2) = 0011_2$

Goal

$f(x) = ((x + 0001_2) \oplus x) >> 0001_2$

Discard

$(\square / x) >> 0001_2$

$\square$	$\mapsto$	$\top\top\top\top$
x	$\mapsto$	1011
$\square / x$	$\mapsto$	$000\top$
$(\square / x) >> 0001_2$	$\mapsto$	0000

Synthesis Algorithm

Generate Components (n <= 1)

Partial Programs

$(\square \oplus x) >> 0001_2 \rightarrow$

$$\begin{aligned} \square &\mapsto 101\top \\ x &\mapsto 1011 \\ \square \oplus x &\mapsto 011\top \\ (\square \oplus x) >> 0001_2 &\mapsto 0011 \end{aligned}$$

Components

$x \ (= 1011_2)$

0001_2

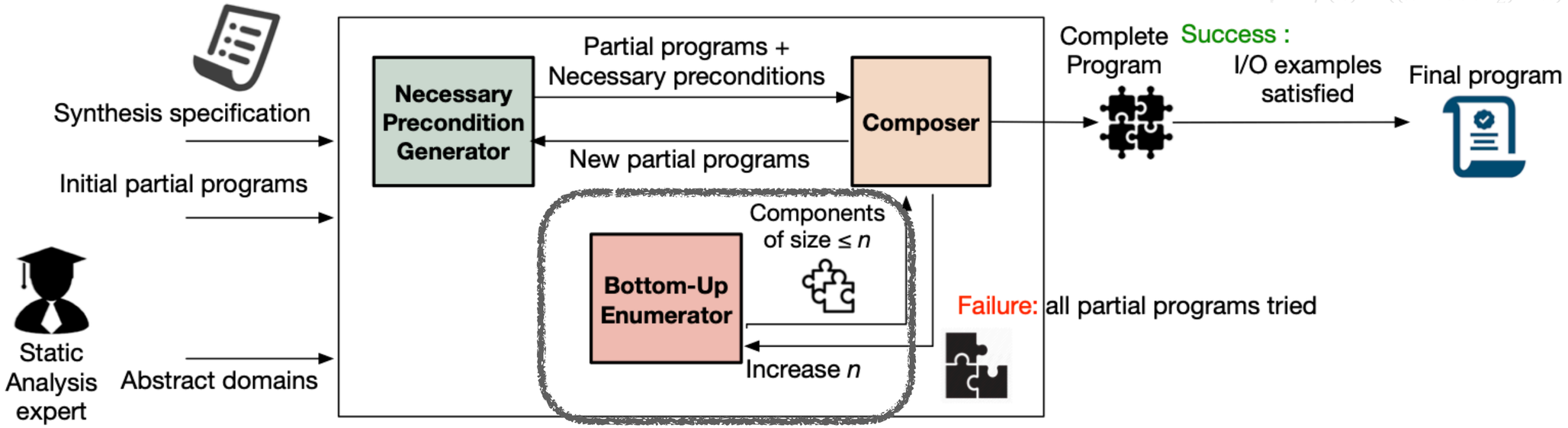
Spec

$$\begin{aligned} S &\rightarrow x \mid 0001_2 \\ &\mid S \wedge S \mid S \vee S \mid S \oplus S \\ &\mid S + S \mid S \times S \mid S / S \mid S >> S \end{aligned}$$

$f(1011_2) = 0011_2$

Goal

$f(x) = ((x + 0001_2) \oplus x) >> 0001_2$



Synthesis Algorithm

Generate Components (n <= 1)

Partial Programs

$(\square \oplus x) >> 0001_2 \rightarrow$

$\square$	$\mapsto$	$110\top$
x	$\mapsto$	1011
$\square \oplus x$	$\mapsto$	$011\top$
$(\square \oplus x) >> 0001_2$	$\mapsto$	0011

Components

$x \ (= 1011_2)$

0001_2

Spec

$$
\begin{aligned}
S &\rightarrow x \mid 0001_2 \\
&\mid S \wedge S \mid S \vee S \mid S \oplus S \\
&\mid S + S \mid S \times S \mid S / S \mid S >> S
\end{aligned}
$$

$f(1011_2) = 0011_2$

Goal

$f(x) = ((x + 0001_2) \oplus x) >> 0001_2$

Synthesis Algorithm

Composition Round 1

Partial Programs

$(\square \oplus x) >> 0001_2 \rightarrow$

$$\begin{array}{rcl} \square & \mapsto & 101\top \\ x & \mapsto & 1011 \\ \square \oplus x & \mapsto & 011\top \\ (\square \oplus x) >> 0001_2 & \mapsto & 0011 \end{array}$$

Components

$x \ (= 1011_2)$

0001_2

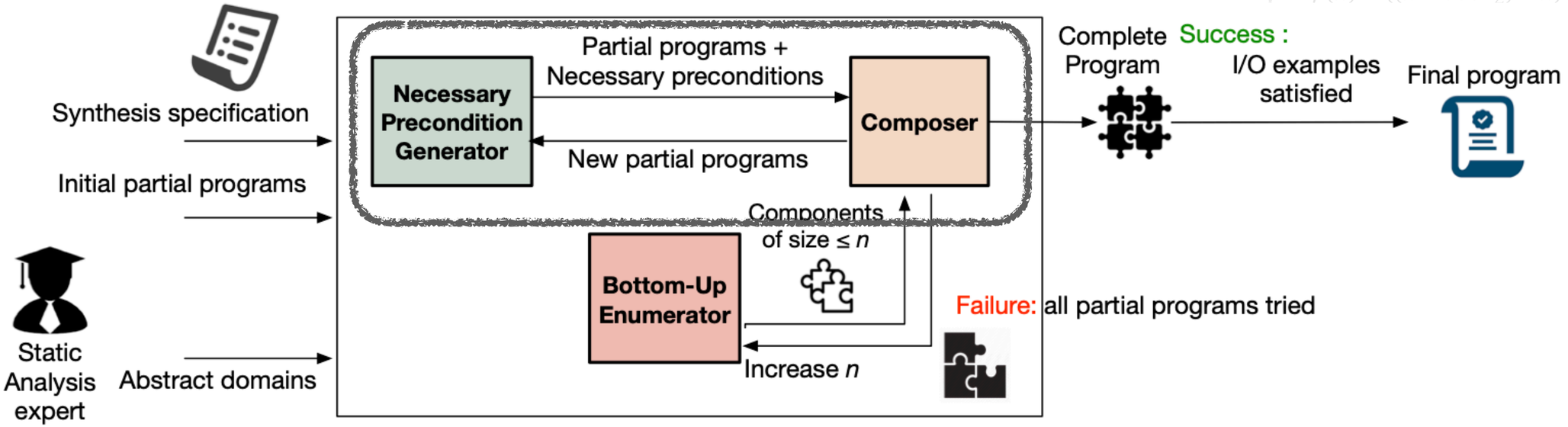
Spec

$$\begin{array}{rcl} S & \rightarrow & x \mid 0001_2 \\ & \mid & S \wedge S \mid S \vee S \mid S \oplus S \\ & \mid & S + S \mid S \times S \mid S / S \mid S >> S \end{array}$$

$f(1011_2) = 0011_2$

Goal

$f(x) = ((x + 0001_2) \oplus x) >> 0001_2$



Synthesis Algorithm

Composition Round 1

Partial Programs

$(\square \oplus x) >> 0001_2 \rightarrow$

$\square$	$\mapsto$	$110\top$
x	$\mapsto$	1011
$\square \oplus x$	$\mapsto$	$011\top$
$(\square \oplus x) >> 0001_2$	$\mapsto$	0011

Components

$x \ (= 1011_2)$

0001_2

No component satisfies precondition -> skip

Spec

$$
\begin{aligned}
S \quad &\rightarrow \quad x \mid 0001_2 \\
&\mid \quad S \wedge S \mid S \vee S \mid S \oplus S \\
&\mid \quad S + S \mid S \times S \mid S / S \mid S >> S
\end{aligned}
$$

$f(1011_2) = 0011_2$

Goal

$f(x) = ((x + 0001_2) \oplus x) >> 0001_2$

Synthesis Algorithm

Generate Components (n <= 3)

Partial Programs

$$(\square \oplus x) >> 0001_2 \rightarrow$$

$$\begin{aligned} \square &\mapsto 101\top \\ x &\mapsto 1011 \\ \square \oplus x &\mapsto 011\top \\ (\square \oplus x) >> 0001_2 &\mapsto 0011 \end{aligned}$$

Components

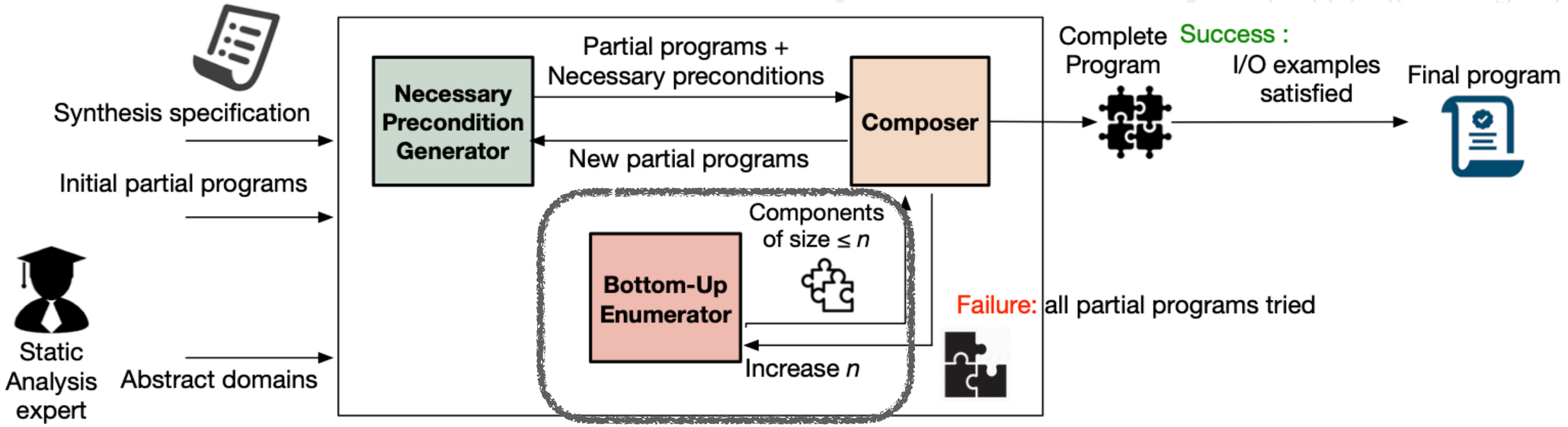
$$x \ (= 1011_2)$$
$$0001_2$$
$$x + 0001_2 \ (= 1100_2)$$
$$x + x \ (= 0110_2)$$
$$x \oplus 0001_2 \ (= 1010_2)$$

Spec

$$\begin{aligned} S &\rightarrow x \mid 0001_2 \\ &\mid S \wedge S \mid S \vee S \mid S \oplus S \\ &\mid S + S \mid S \times S \mid S / S \mid S >> S \end{aligned}$$
$$f(1011_2) = 0011_2$$

Goal

$$f(x) = ((x + 0001_2) \oplus x) >> 0001_2$$



Synthesis Algorithm

Generate Components (n <= 3)

Partial Programs

$(\square \oplus x) >> 0001_2 \rightarrow$

$\square$	$\mapsto$	$110\top$
x	$\mapsto$	1011
$\square \oplus x$	$\mapsto$	$011\top$
$(\square \oplus x) >> 0001_2$	$\mapsto$	0011

Components

$x \ (= 1011_2)$

0001_2

$x + 0001_2 \ (= 1100_2)$

$x + x \ (= 0110_2)$

$x \oplus 0001_2 \ (= 1010_2)$

...

Size increased

Spec

$$
\begin{aligned}
S &\rightarrow x \mid 0001_2 \\
&\mid S \wedge S \mid S \vee S \mid S \oplus S \\
&\mid S + S \mid S \times S \mid S / S \mid S >> S
\end{aligned}
$$

$f(1011_2) = 0011_2$

Goal

$f(x) = ((x + 0001_2) \oplus x) >> 0001_2$

Synthesis Algorithm

Generate Components (n <= 3)

Partial Programs

$(\square \oplus x) >> 0001_2 \rightarrow$

$\square$	$\mapsto$	$110\top$
x	$\mapsto$	1011
$\square \oplus x$	$\mapsto$	$011\top$
$(\square \oplus x) >> 0001_2$	$\mapsto$	0011

Components

$x \ (= 1011_2)$

0001_2

$x + 0001_2 \ (= 1100_2)$

$x + x \ (= 0110_2)$

$x \oplus 0001_2 \ (= 1010_2)$

...

Spec

$S \rightarrow x \mid 0001_2$

$\mid S \wedge S \mid S \vee S \mid S \oplus S$

$\mid S + S \mid S \times S \mid S / S \mid S >> S$

$f(1011_2) = 0011_2$

Goal

$f(x) = ((x + 0001_2) \oplus x) >> 0001_2$

~~$x/1 \ (= 1011_2)$~~

$x \wedge x \ (= 1011_2)$

~~$x \wedge 0001_2 \ (= 1011_2)$~~

~~$x / x \ (= 0001_2)$~~

~~$0001_2 \vee 0001_2 \ (= 0001_2)$~~

Ignore Observational Equivalent Components

Synthesis Algorithm

Composition Round 2

Partial Programs

$(\square \oplus x) >> 0001_2 \rightarrow$

$\square$	$\mapsto$	$110\top$
x	$\mapsto$	1011
$\square \oplus x$	$\mapsto$	$011\top$
$(\square \oplus x) >> 0001_2$	$\mapsto$	0011

Components

$x \ (= 1011_2)$

0001_2

$x + 0001_2 \ (= 1100_2)$

$x + x \ (= 0110_2)$

$x \oplus 0001_2 \ (= 1010_2)$

...

Spec

$$
\begin{aligned}
S \quad &\rightarrow \quad x \mid 0001_2 \\
&\mid \quad S \wedge S \mid S \vee S \mid S \oplus S \\
&\mid \quad S + S \mid S \times S \mid S / S \mid S >> S
\end{aligned}
$$

$f(1011_2) = 0011_2$

Goal

$f(x) = ((x + 0001_2) \oplus x) >> 0001_2$

Synthesis Algorithm

Composition Round 2

Partial Programs

$(\square \oplus x) >> 0001_2 \rightarrow$

$\square$	$\mapsto$	$110\top$
x	$\mapsto$	1011
$\square \oplus x$	$\mapsto$	$011\top$
$(\square \oplus x) >> 0001_2$	$\mapsto$	0011

$\{1100, 1101\} \rightarrow$

Components

$x \ (= 1011_2)$

0001_2

$x + 0001_2 \ (= 1100_2)$

$x + x \ (= 0110_2)$

$x \oplus 0001_2 \ (= 1010_2)$

...

Lookup Table

0001	:	0001_2
0110	:	$x + x$
1010	:	$x \oplus 0001_2$
1011	:	x
1100	:	$x + 0001_2$

Spec

$$
\begin{aligned}
S \ &\rightarrow \ x \mid 0001_2 \\
&\mid \ S \wedge S \mid S \vee S \mid S \oplus S \\
&\mid \ S + S \mid S \times S \mid S / S \mid S >> S
\end{aligned}
$$

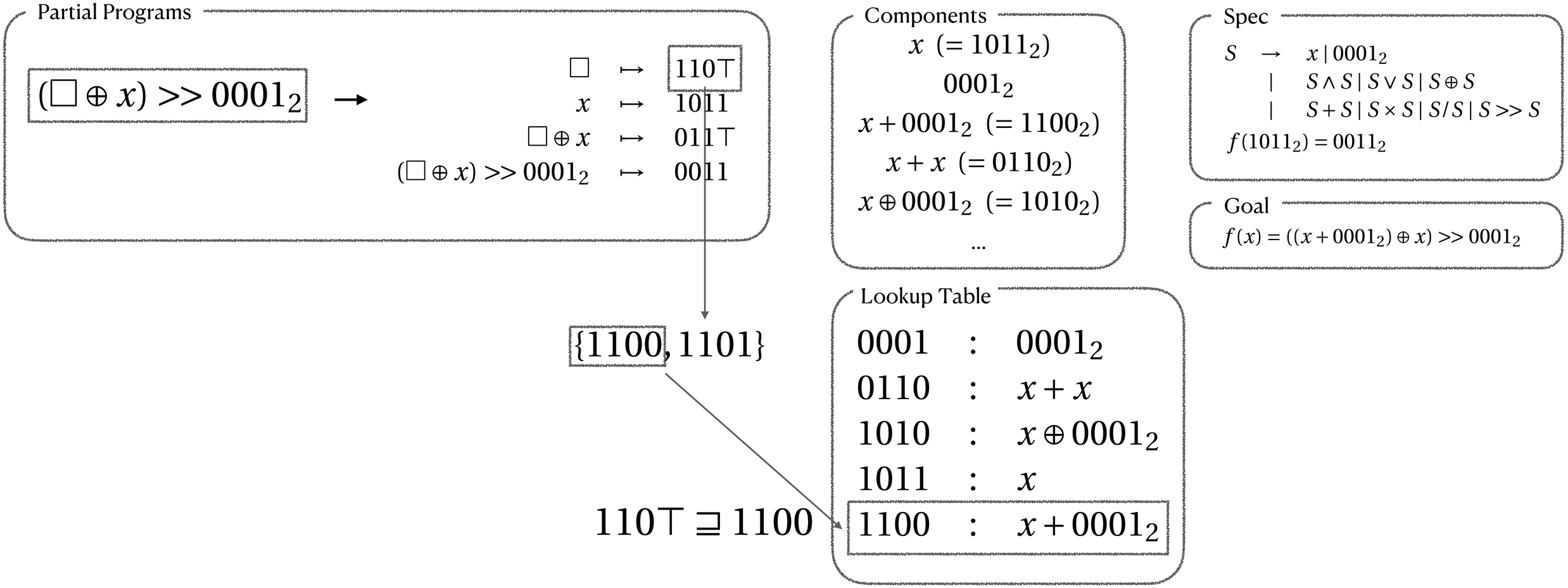
$f(1011_2) = 0011_2$

Goal

$f(x) = ((x + 0001_2) \oplus x) >> 0001_2$

Synthesis Algorithm

Composition Round 2



Synthesis Algorithm

Found Solution

Partial Programs

$(\square \oplus x) >> 0001_2$ →

$\square$	$\mapsto$	$110\top$
x	$\mapsto$	1011
$\square \oplus x$	$\mapsto$	$011\top$
$(\square \oplus x) >> 0001_2$	$\mapsto$	0011

Components

$x \ (= 1011_2)$

0001_2

$x + 0001_2 \ (= 1100_2)$

$x + x \ (= 0110_2)$

$x \oplus 0001_2 \ (= 1010_2)$

...

Spec

$$
\begin{aligned}
S &\rightarrow x \mid 0001_2 \\
&\mid S \wedge S \mid S \vee S \mid S \oplus S \\
&\mid S + S \mid S \times S \mid S / S \mid S >> S
\end{aligned}
$$

$f(1011_2) = 0011_2$

Goal

$f(x) = ((x + 0001_2) \oplus x) >> 0001_2$

New Candidate Solution

$f(x) = ((x + 0001_2) \oplus x) >> 0001_2$

Solution Found!

$$
\begin{aligned}
f(1011_2) &= ((1011_2 + 0001_2) \oplus 1011_2) >> 0001_2 \\
&= (1100_2 \oplus 1011_2) >> 0001_2 \\
&= 0111_2 >> 0001_2 \\
&= 0011_2
\end{aligned}
$$

Synthesizing Conditional Programs

Simba Core + Divde-and-Conquer

- Incorporate the divide-and-conquer approach [Alur et al. 2017] into our algorithm

Syntactic Spec

$$
\begin{array}{rcl}
S & \rightarrow & x \mid 0000_2 \mid 0001_2 \\
 & \mid & S + S \mid \cdots \\
 & \mid & \boxed{\text{if } S \text{ then } S \text{ else } S}
\end{array}
$$

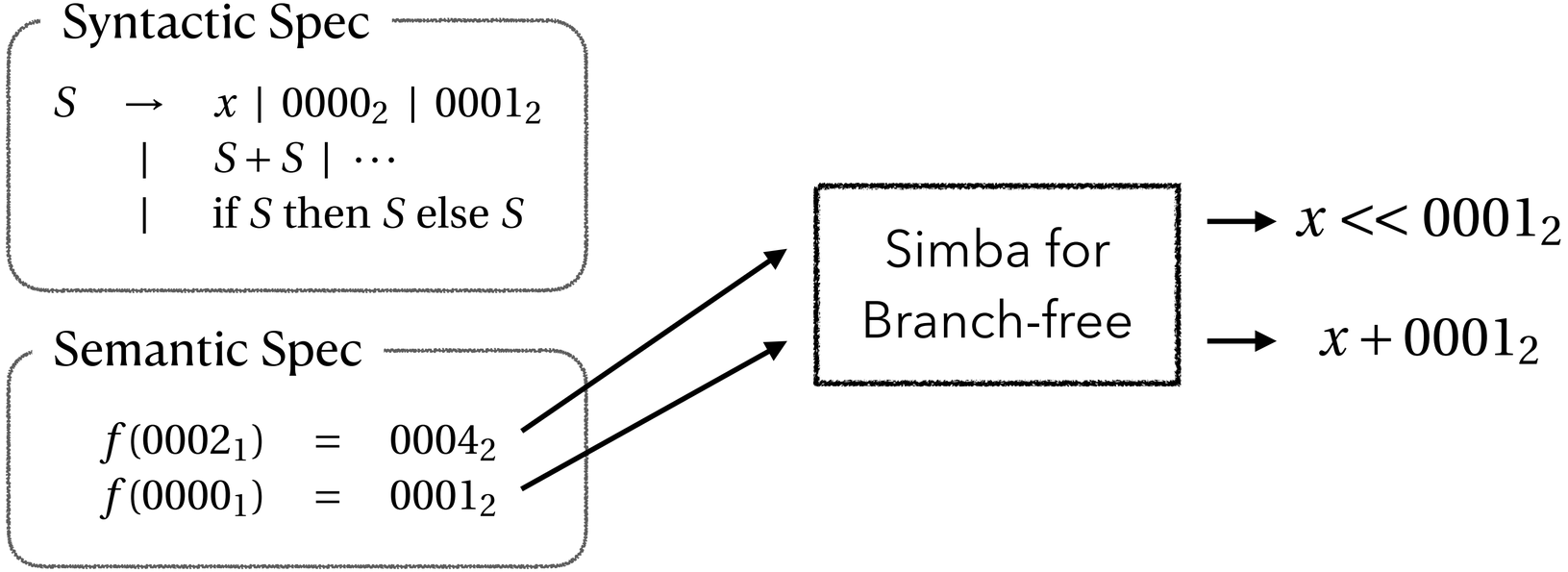
Semantic Spec

$$
\begin{array}{rcl}
f(0002_1) & = & 0004_2 \\
f(0000_1) & = & 0001_2
\end{array}
$$

Synthesizing Conditional Programs

Simba Core + Divde-and-Conquer

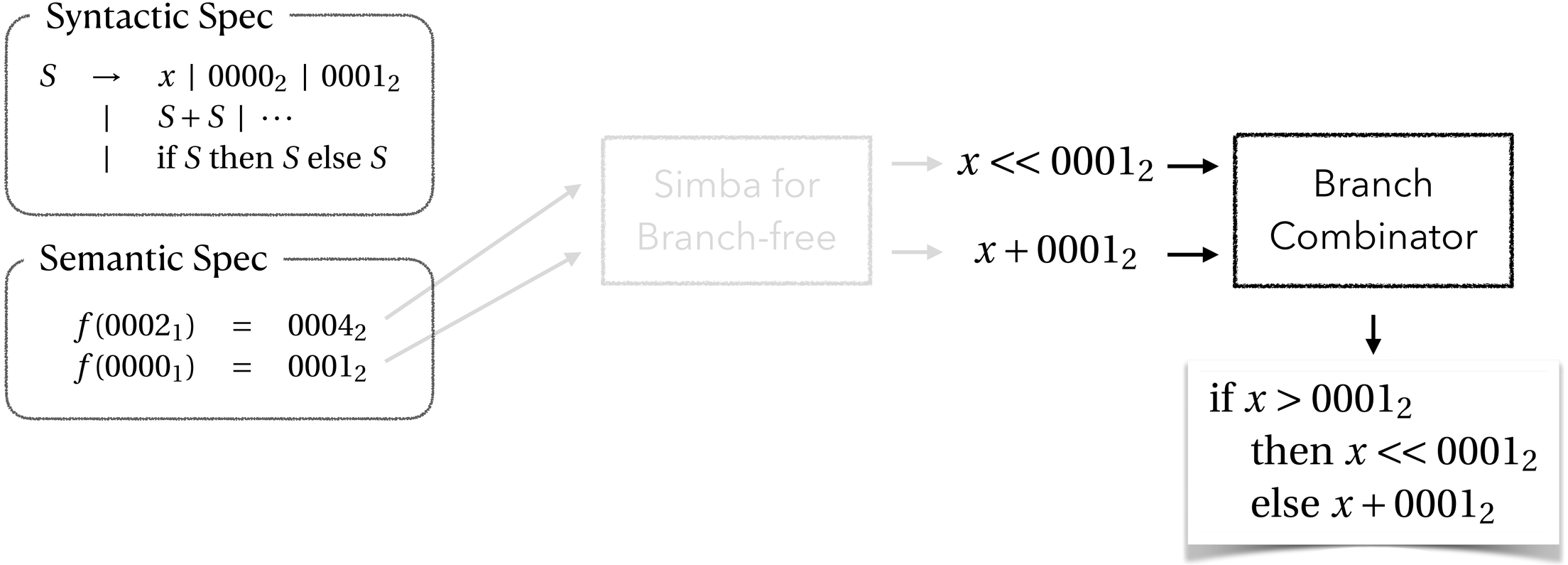
- Incorporate the divide-and-conquer approach [Alur et al. 2017] into our algorithm



Synthesizing Conditional Programs

Simba Core + Divde-and-Conquer

- Incorporate the divide-and-conquer approach [Alur et al. 2017] into our algorithm



Evaluation: Benchmarks

- 1875 synthesis tasks from 3 domains: BitVec, BitVec-Cond, Circuit

Background Theory	**Bit-vector arithmetic**			**SAT**	
Domain	**BitVec (# 544)**		**BitVec-Cond (# 750)**	**Circiut (# 581)**	
Benchmarks	HD	Deobfusc	-	Lobster	Crypto
# Tasks	44	500	750	369	212

Evaluation: Benchmarks

- 1875 synthesis tasks from 3 domains: BitVec, BitVec-Cond, Circuit

Background Theory	**Bit-vector arithmetic**			**SAT**	
Domain	**BitVec (# 544)**		**BitVec-Cond (# 750)**	**Circiut (# 581)**	
Benchmarks	HD	Deobfusc	-	Lobster	Crypto
# Tasks	44	500	750	369	212

Evaluation: Benchmarks

- 1875 synthesis tasks from 3 domains: BitVec, BitVec-Cond, Circuit

Background Theory	**Bit-vector arithmetic**			**SAT**	
Domain	**BitVec (# 544)**		**BitVec-Cond (# 750)**	**Circiut (# 581)**	
Benchmarks	HD	Deobfusc	-	Lobster	Crypto
# Tasks	44	500	750	369	212

Evaluation: Benchmarks

- 1875 synthesis tasks from 3 domains: BitVec, BitVec-Cond, Circuit

Background Theory	**Bit-vector arithmetic**			**SAT**	
Domain	**BitVec (# 544)**		**BitVec-Cond (# 750)**	**Circiut (# 581)**	
Benchmarks	HD	Deobfusc	-	Lobster	Crypto
# Tasks	44	500	750	369	212

Evaluation: Benchmarks

- 1875 synthesis tasks from 3 domains: BitVec, BitVec-Cond, Circuit

Background Theory	**Bit-vector arithmetic**			**SAT**	
Domain	**BitVec (# 544)**		**BitVec-Cond (# 750)**	**Circiut (# 581)**	
Benchmarks	HD	Deobfusc	-	Lobster	Crypto
# Tasks	44	500	750	369	212

Hacker's **D**elight
from SyGuS competition

SyGuS competition PBE-**BitVec**tor
(may contain **cond**itionals)

Circuit transformation
related to **crypto**graphic modules
from SyGus competition

Evaluation: Benchmarks

- 1875 synthesis tasks from 3 domains: BitVec, BitVec-Cond, Circuit

Background Theory	Bit-vector arithmetic			SAT	
Domain	**BitVec (# 544)**		**BitVec-Cond (# 750)**	**Circiut (# 581)**	
Benchmarks	HD	Deobfusc	-	Lobster	Crypto
# Tasks	44	500	750	369	212

Hacker's **D**elight
from SyGuS competition

Deobfuscator
from QSynth [David et al. 2020]

SyGuS competition PBE-**BitVec**tor
(may contain **cond**itionals)

Circuit transformation
from **Lobster** [Lee et al. 2020]

Circuit transformation
related to **crypto**graphic modules
from SyGus competition

Evaluation: Baseline Solvers

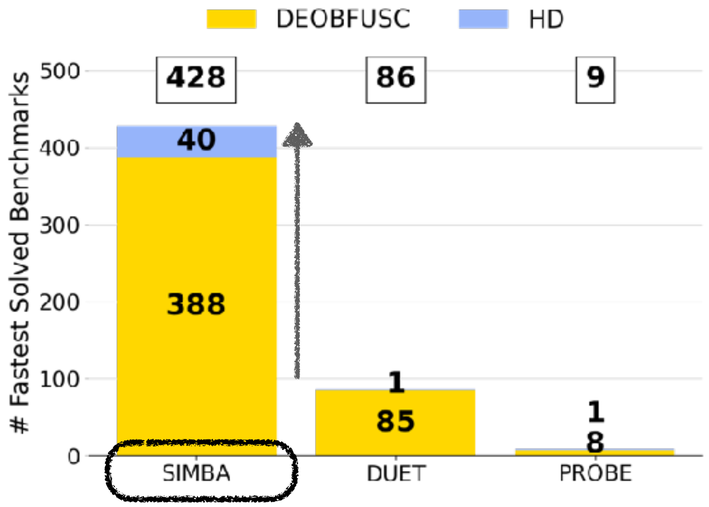
- Duet [Lee 2021]: the state-of-the-art synthesis tool for inductive SyGuS problems
  - Employs a bidirectional(top-down + bottom-up) search strategy
  - Requires *inverse semantics operators* for search space pruning

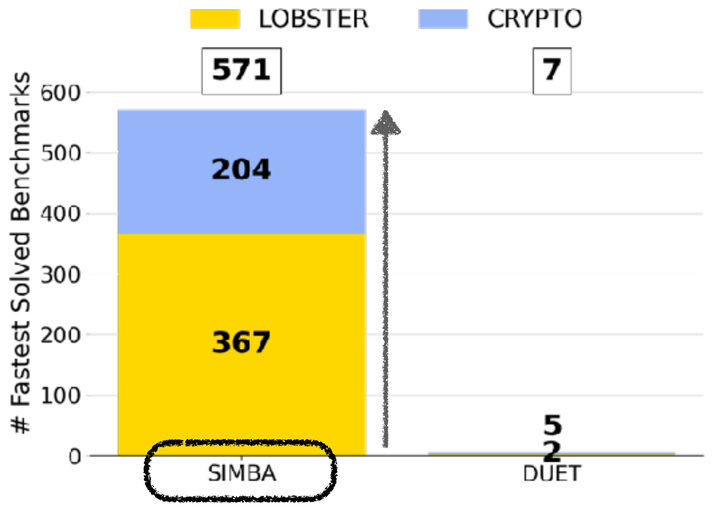
- Probe [Barke et al. 2020]: tool performs a bottom-up search with probabilistic model
  - Probabilistic model is learned *just in time* during the search processes

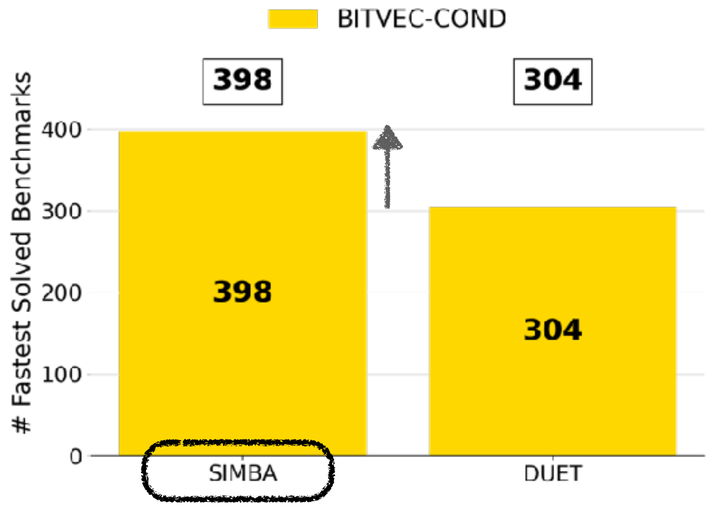
Evaluation: Simba Performance

Solve Faster

- Outperformed baseline solvers for conditional-free program
- Comparable to baseline solvers for conditional program





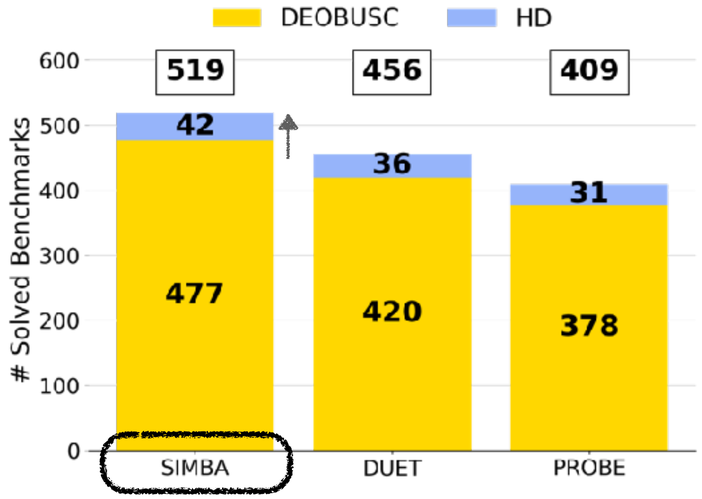


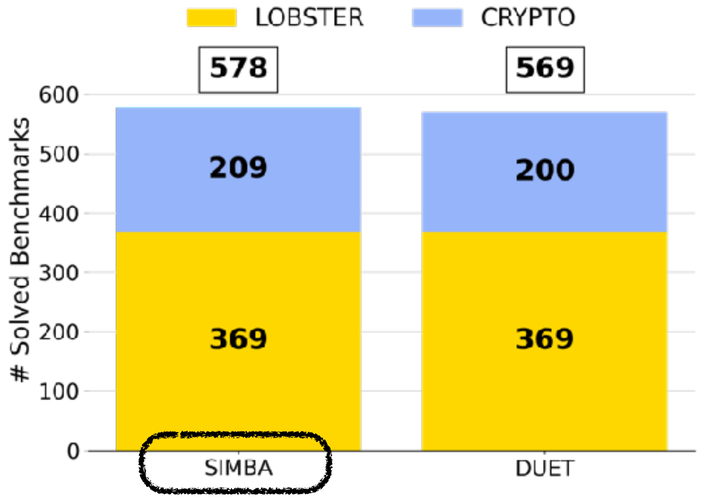
of the fastest-solved problems for each domain

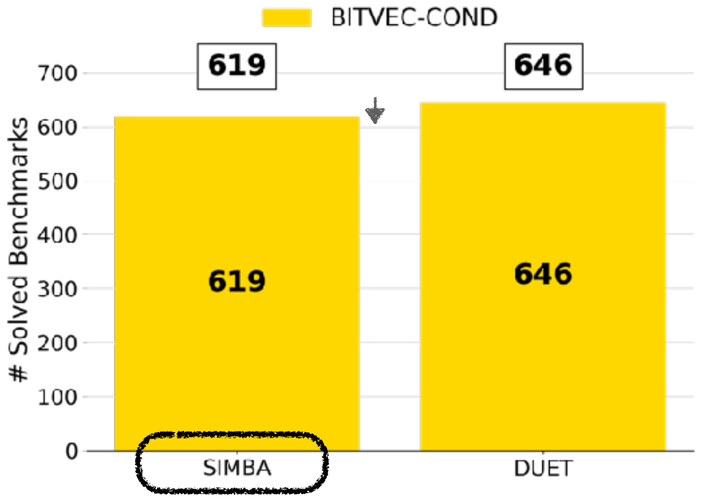
Evaluation: Simba Performance

Solve More

- Outperformed baseline solvers for conditional-free program
- Comparable to baseline solvers for conditional program





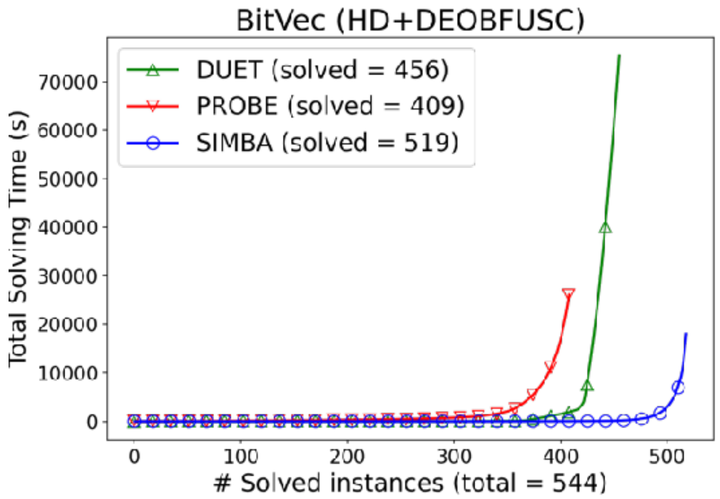


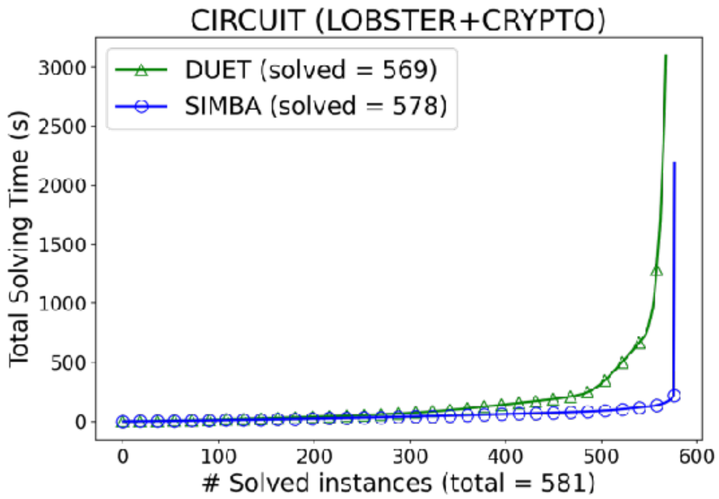
of the solved problems with 1h timeout for each domain

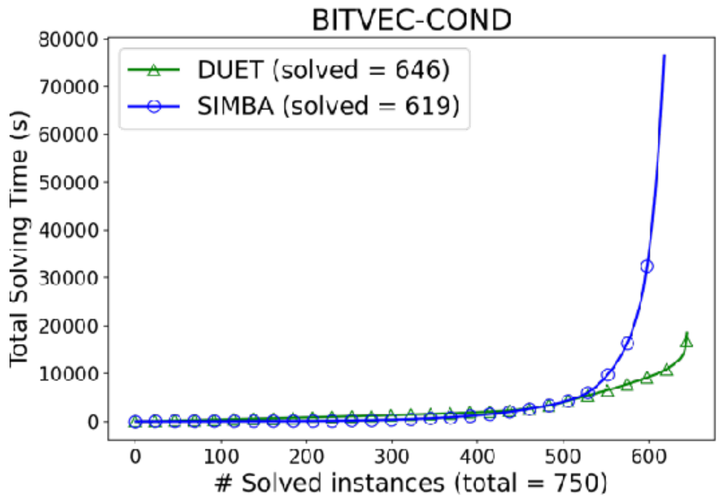
Evaluation: Simba Performance

Overall

- SIMBA(plot ○) shows the best performance for branch-free benchmarks
- DUET(plot △) is better than SIMBA for branch benchmarks in average





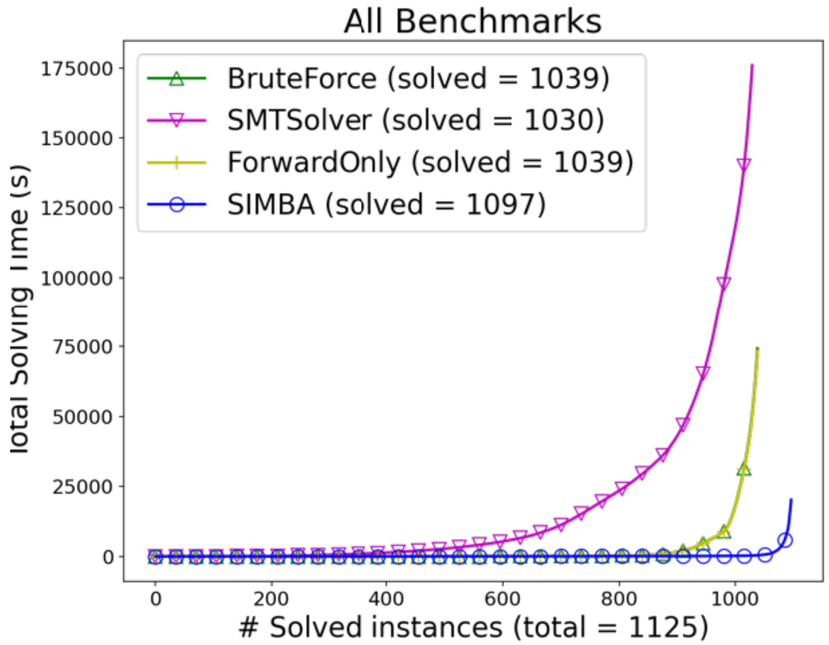


Cactus-plots

Evaluation: Simba Performance

Ablation Study

- Forward-Backward Analysis(plot ○) is necessary to achieve the best performance
- Only forward analysis(plot +) or SMT Solver(plot ▽) is not sufficient
  - Sometimes, even worse than brute force(plot △) because of bad analysis cost



In the paper...

- Formalization (abstract domain, forward-backward transfer functions, etc.)
- Detailed Algorithm of synthesis and forward-backward analysis
- Why our tool outperforms the baseline solvers (case study)
- and more

⬇Paper

😊 Thank You!

GitHub Repo⬇