Symbolic Optimization with SMT Solvers

Aws Albarghouthi / UToronto
Marsha Chechik / UToronto
Arie Gurfinkel / CMU
Zachary Kincaid / UToronto
Yi Li / UToronto

POPL 2014 / San Diego, CA

SMT Explosion!

SMT solvers appear everywhere. Why?

- *Amazing performance!*
- *Support a large range of logical theories*
- *We've become really good at casting problems as SMT queries!*

SMT Applications

Verification

- *Checking VCs, invariant generation, etc.*

Bug finding

- *Symbolic execution, BMC, fuzzing, etc.*

Synthesis

- *Circuit synthesis, sketching, superoptimization, etc.*

Functional programming

- *Liquid types*

SMT Applications

Verification

- *Checking VCs, invariant generation, etc.*

Bug finding

- *Symbolic execution, BMC, fuzzing, etc.*

Synthesis

- *Circuit synthesis, sketching, superoptimization, etc.*

Functional programming

- *Liquid types*

~22% of POPL'14 papers mention SMT solvers!

How are SMT Solvers Used?

Finding models

- *Bug finding*: *erroneous traces*
- *Synthesis*: *program/circuit*

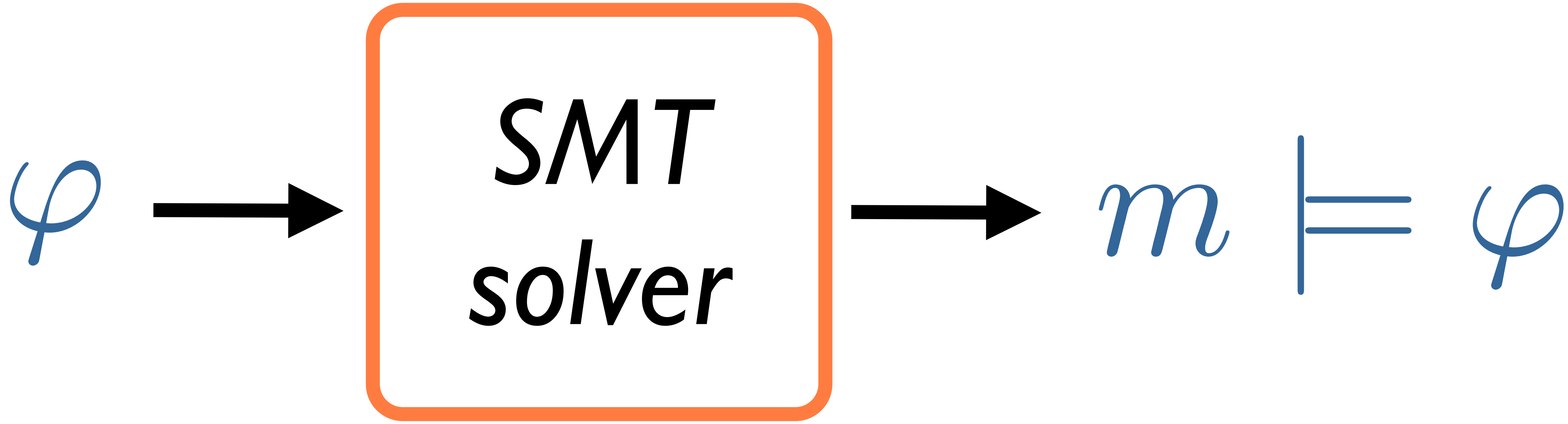
Proving unsatisfiability (validity)

- *Verification*: *VC holds*
- *Refinement types*: *subtyping relation holds*

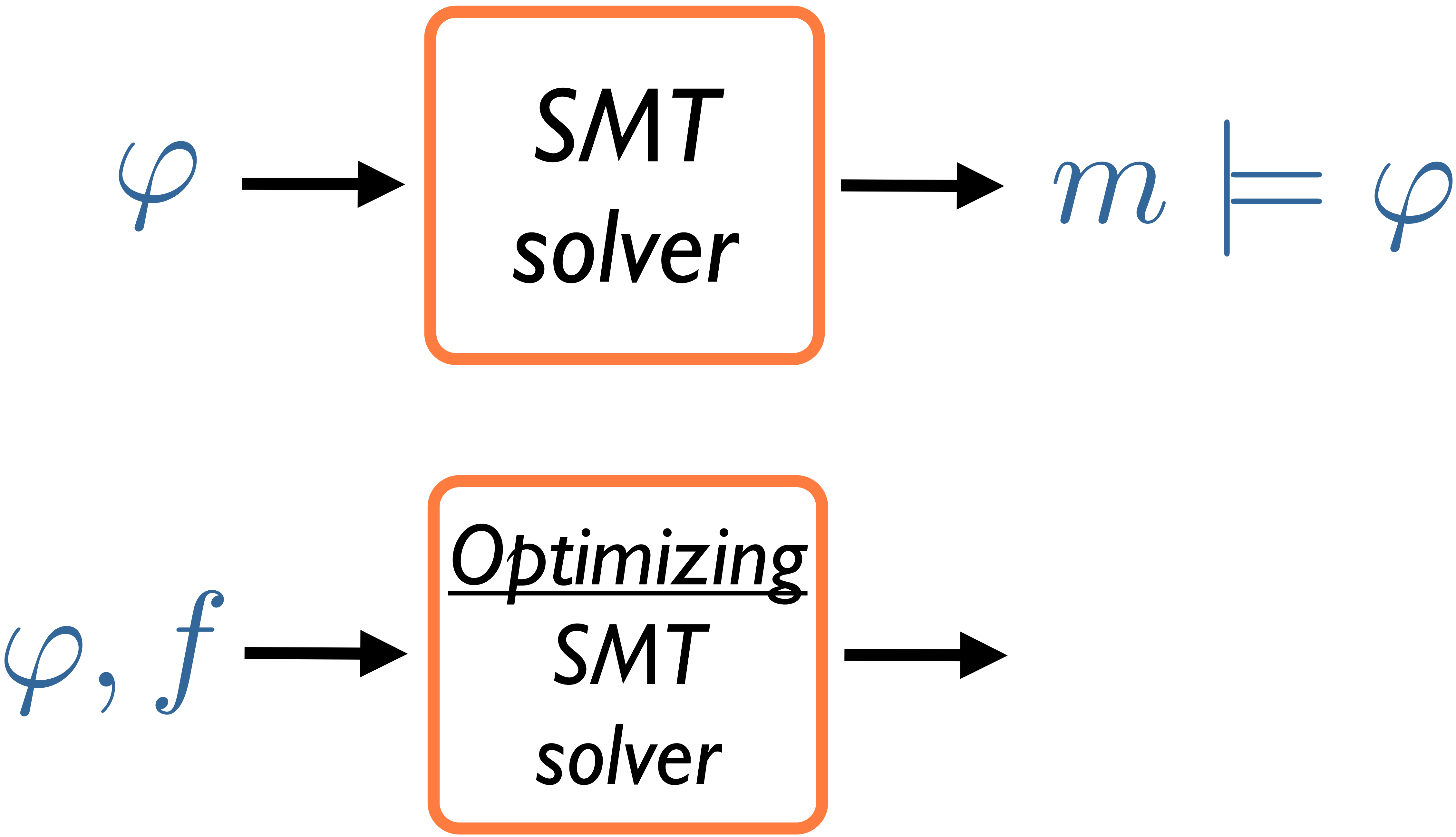
How are SMT Solvers Used?

What about <u>optimization</u>?

Optimal Models



Optimal Models



Optimal Models

$$\varphi \longrightarrow \boxed{\textit{SMT solver}} \longrightarrow m \models \varphi$$

$$\varphi, f \longrightarrow \boxed{\begin{array}{c}\underline{\textit{Optimizing}}\\ \textit{SMT solver}\end{array}} \longrightarrow \begin{array}{c} m \models \varphi \\ \max\ f(m) \end{array}$$

Why Should You Care?

Plenty of applications for optimization:

- *Numerical invariant generation*
- *Counterexample generation*
- *Program synthesis*
- *Constraint programming*
- *... and many others*

Problem Statement

$$\varphi \in \mathcal{T} \cup \mathcal{LRA}$$

signature disjoint

Problem Statement

$\varphi \in \mathcal{T} \cup \mathcal{LRA}$ *signature disjoint*	*E.g.:* $3x + 2y \leq 0 \vee z \geq 3$

Problem Statement

$$\varphi \in \mathcal{T} \cup \mathcal{LRA}$$

signature disjoint

E.g.: $3x + 2y \leq 0 \vee z \geq 3$

Set of linear *objective functions*: $f_1, \ldots, f_n$

E.g.: $x + 2y,\, z$

Problem Statement

$$\varphi \in \mathcal{T} \cup \mathcal{LRA}$$

signature disjoint

E.g.: $3x + 2y \leq 0 \vee$
$z \geq 3$

Set of linear *objective functions:* $f_1, \ldots, f_n$

E.g.: $x + 2y, z$

Goal: find assignments $m_1, \ldots, m_n$

$$m_1 \models \varphi \; s.t. \; \max \; f_1(m_1)$$
$$\ldots$$
$$m_n \models \varphi \; s.t. \; \max \; f_n(m_n)$$

Problem Statement

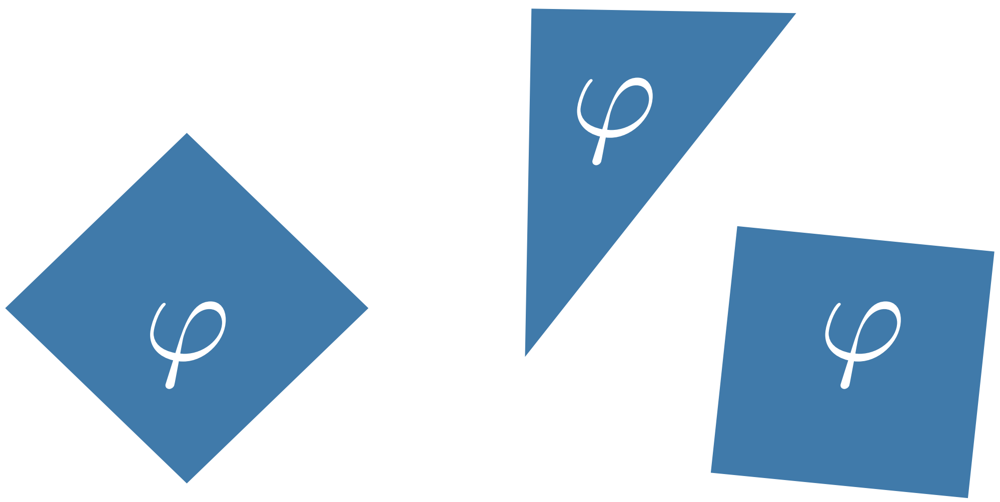
$$\varphi \in \mathcal{T} \cup \mathcal{LRA}$$

Set of linear *objective functions*: $f_1, \ldots, f_n$

Problem Statement

$$\varphi \in \mathcal{T} \cup \mathcal{LRA}$$

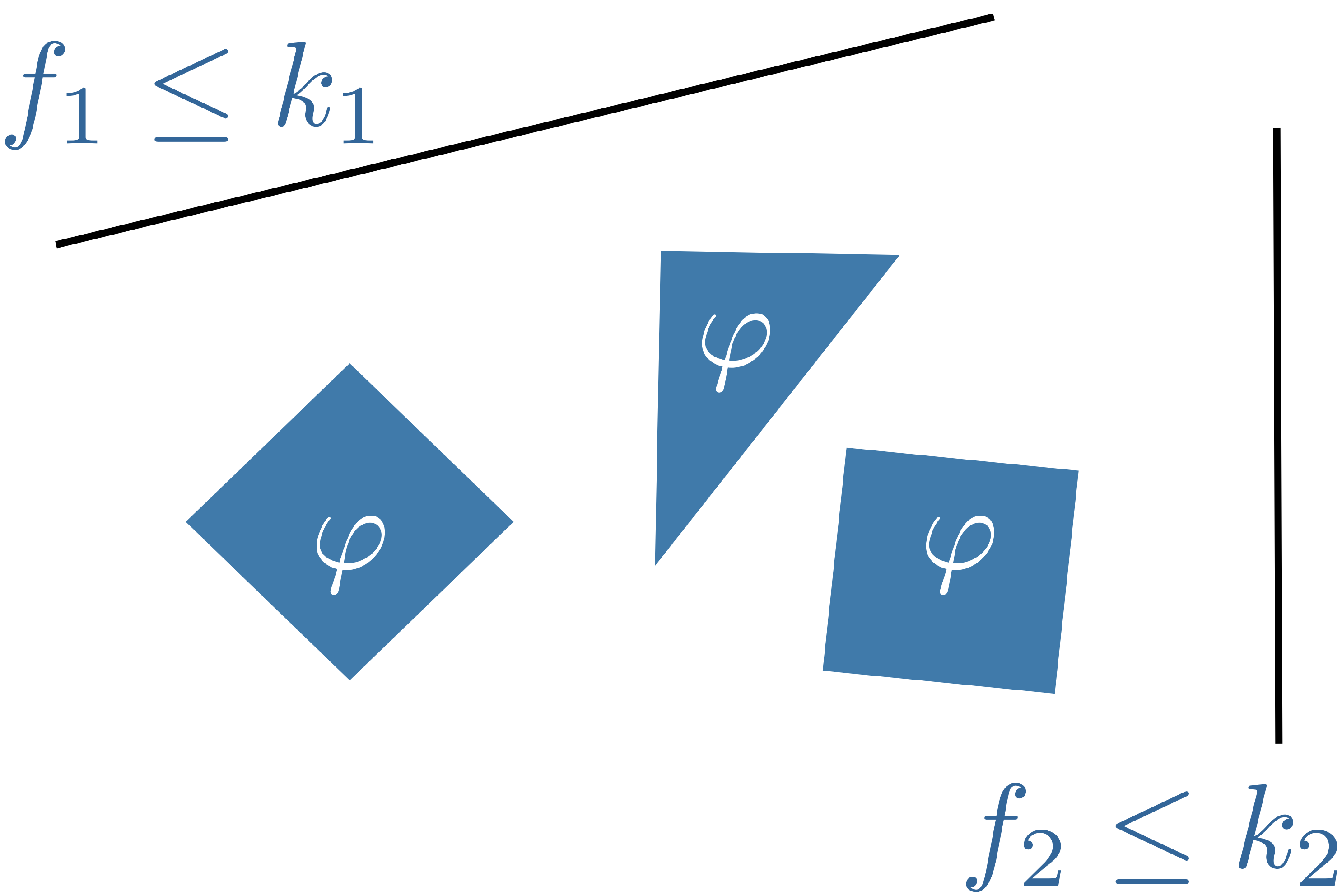
Set of linear <u>*objective functions:*</u> $f_1, \ldots, f_n$



Problem Statement

$$\varphi \in \mathcal{T} \cup \mathcal{LRA}$$

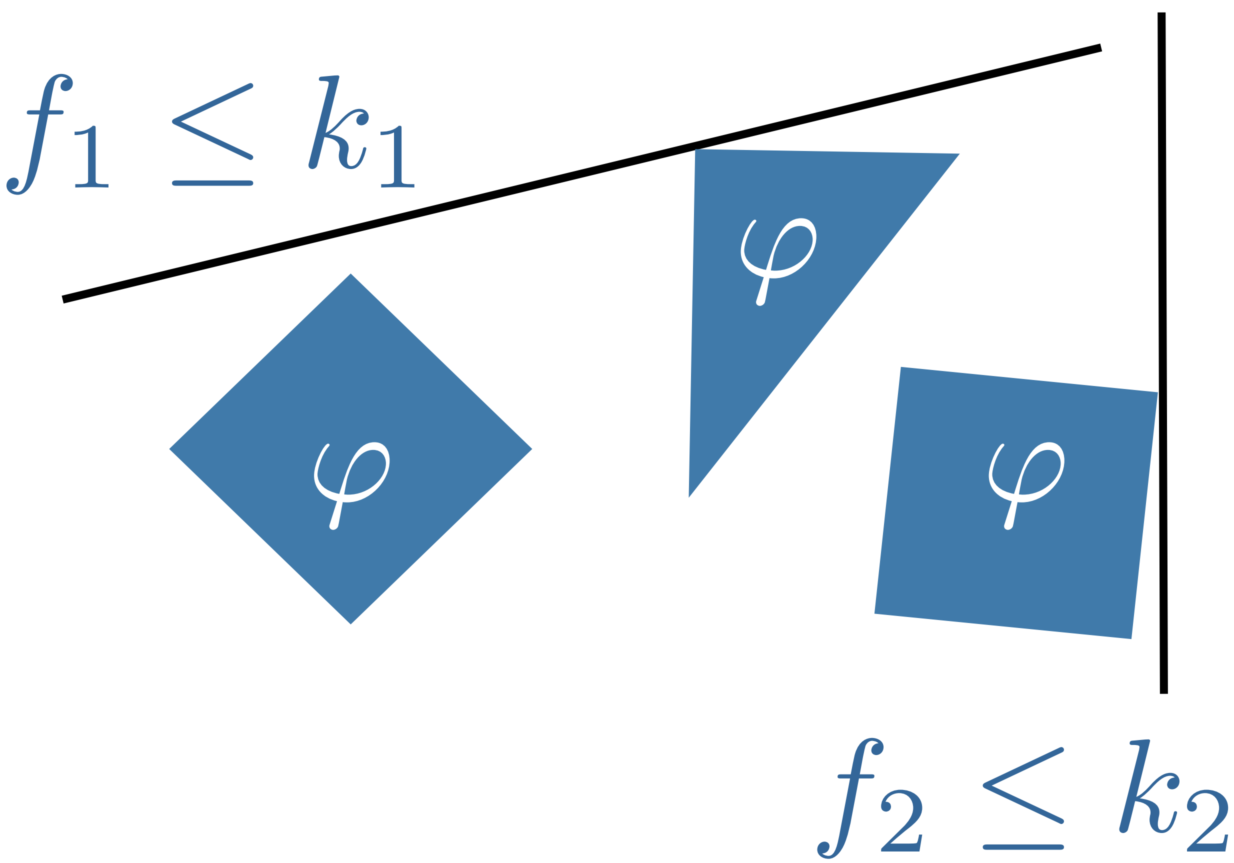
Set of linear *objective functions*: $f_1, \ldots, f_n$



Problem Statement

$$\varphi \in \mathcal{T} \cup \mathcal{LRA}$$

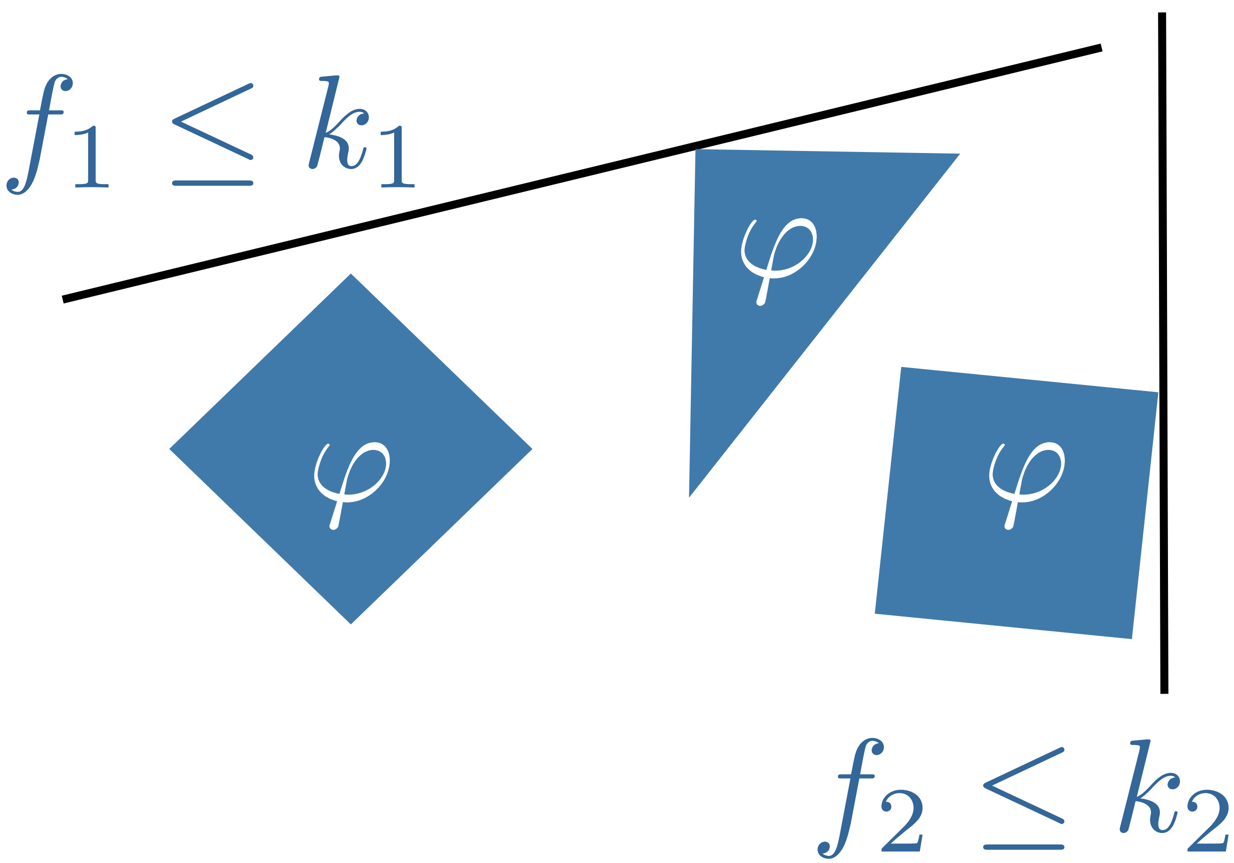
Set of linear *objective functions*: $f_1, \ldots, f_n$



Problem Statement

$$\varphi \in \mathcal{T} \cup \mathcal{LRA}$$

Set of linear *objective functions:* $f_1, \ldots, f_n$



Find strongest

$$\bigwedge_{i \in [1,n]} f_i \le k_i$$

that contains φ

Challenges & Contributions

Symba: an SMT-based optimization algorithm

- *Non-convex optimization*
- *Linear arithmetic modulo theories*
- *Multiple independent objectives*
- *SMT solver as a black box*

Outline

Symba by example

Application and evaluation

What’s next?

Example

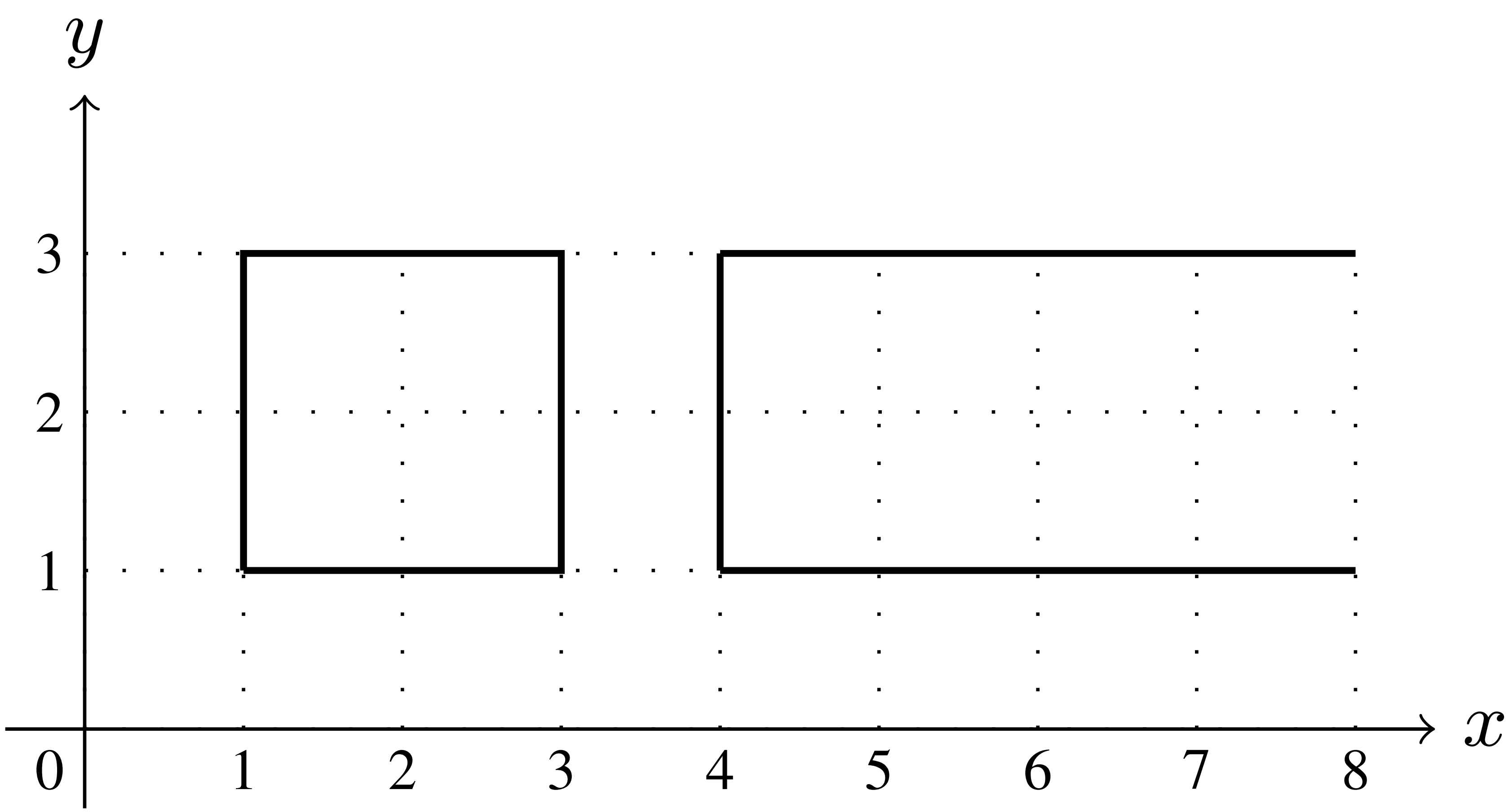
$$\varphi \equiv 1 \leq y \leq 3 \wedge (1 \leq x \leq 3 \vee x \geq 4)$$

Objective functions: $\{y, x + y\}$

Example

$$\varphi \equiv 1 \leq y \leq 3 \wedge (1 \leq x \leq 3 \vee x \geq 4)$$

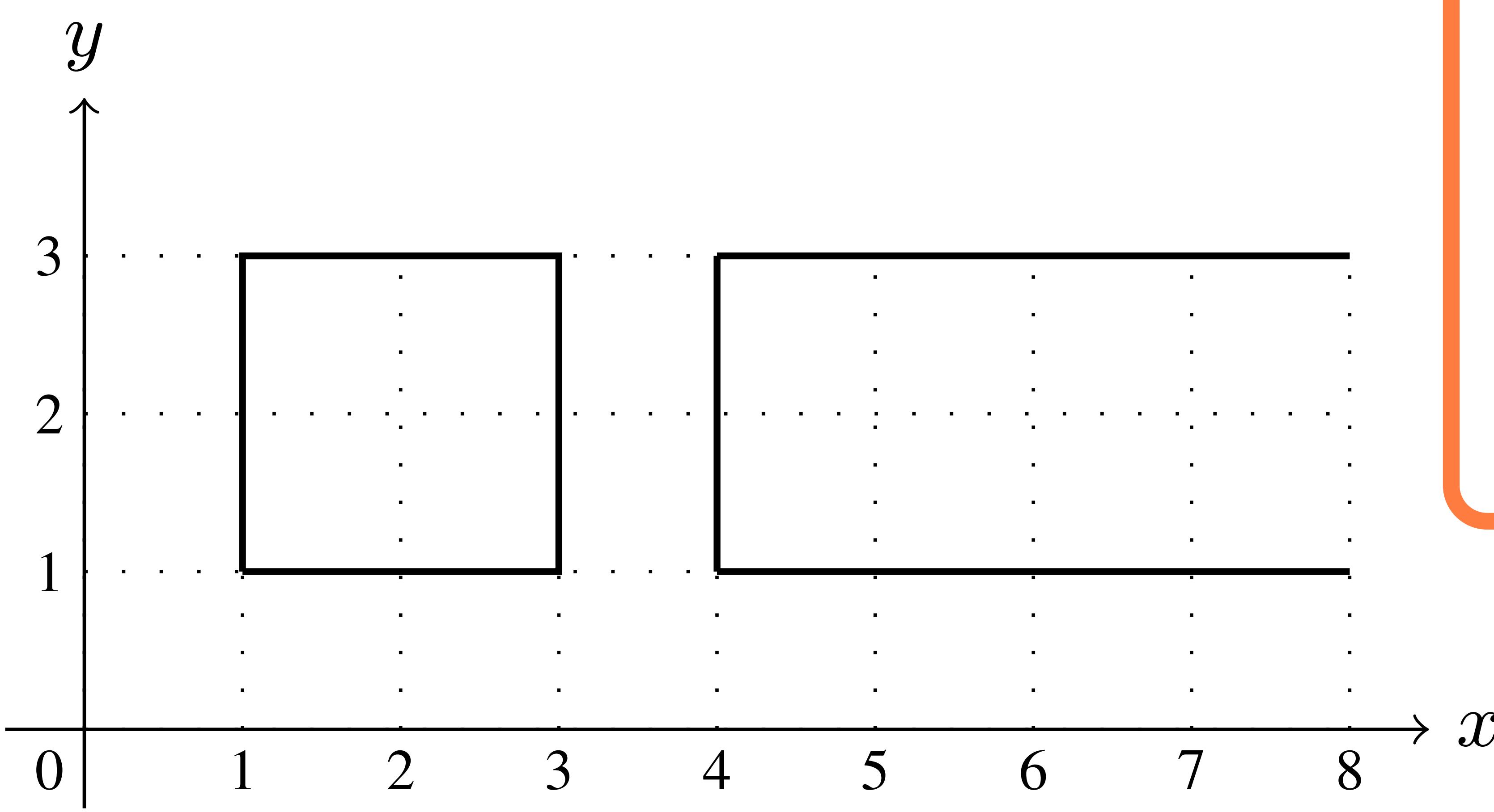
Objective functions: $\{y, x + y\}$



Example

$$\varphi \equiv 1 \leq y \leq 3 \wedge (1 \leq x \leq 3 \vee x \geq 4)$$

Objective functions: $\{y, x + y\}$



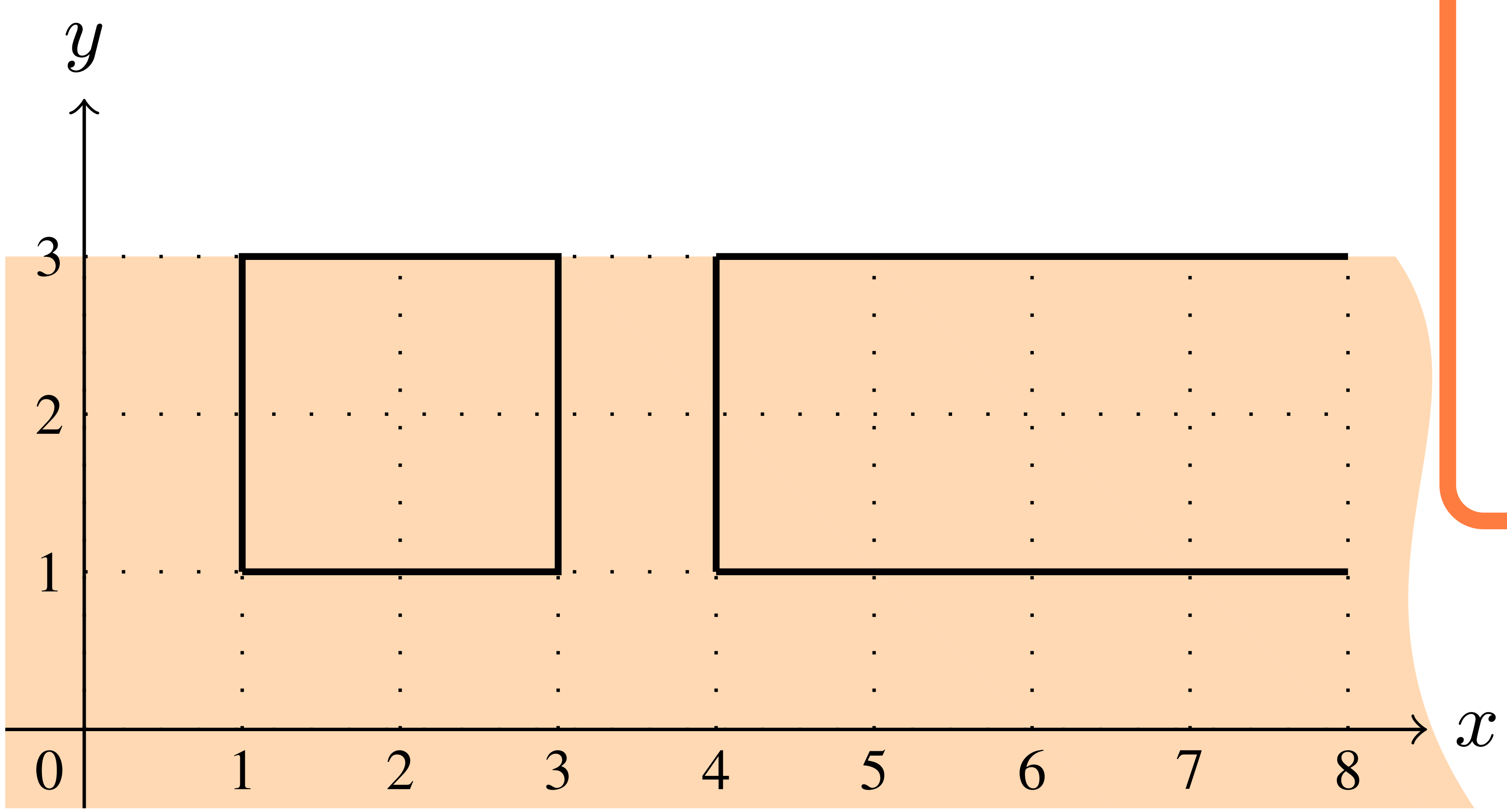
Optimal Solution:

$$y \leq 3 \wedge$$
$$x + y \leq \infty$$

Example

$$\varphi \equiv 1 \leq y \leq 3 \wedge (1 \leq x \leq 3 \vee x \geq 4)$$

Objective functions: $\{y, x + y\}$



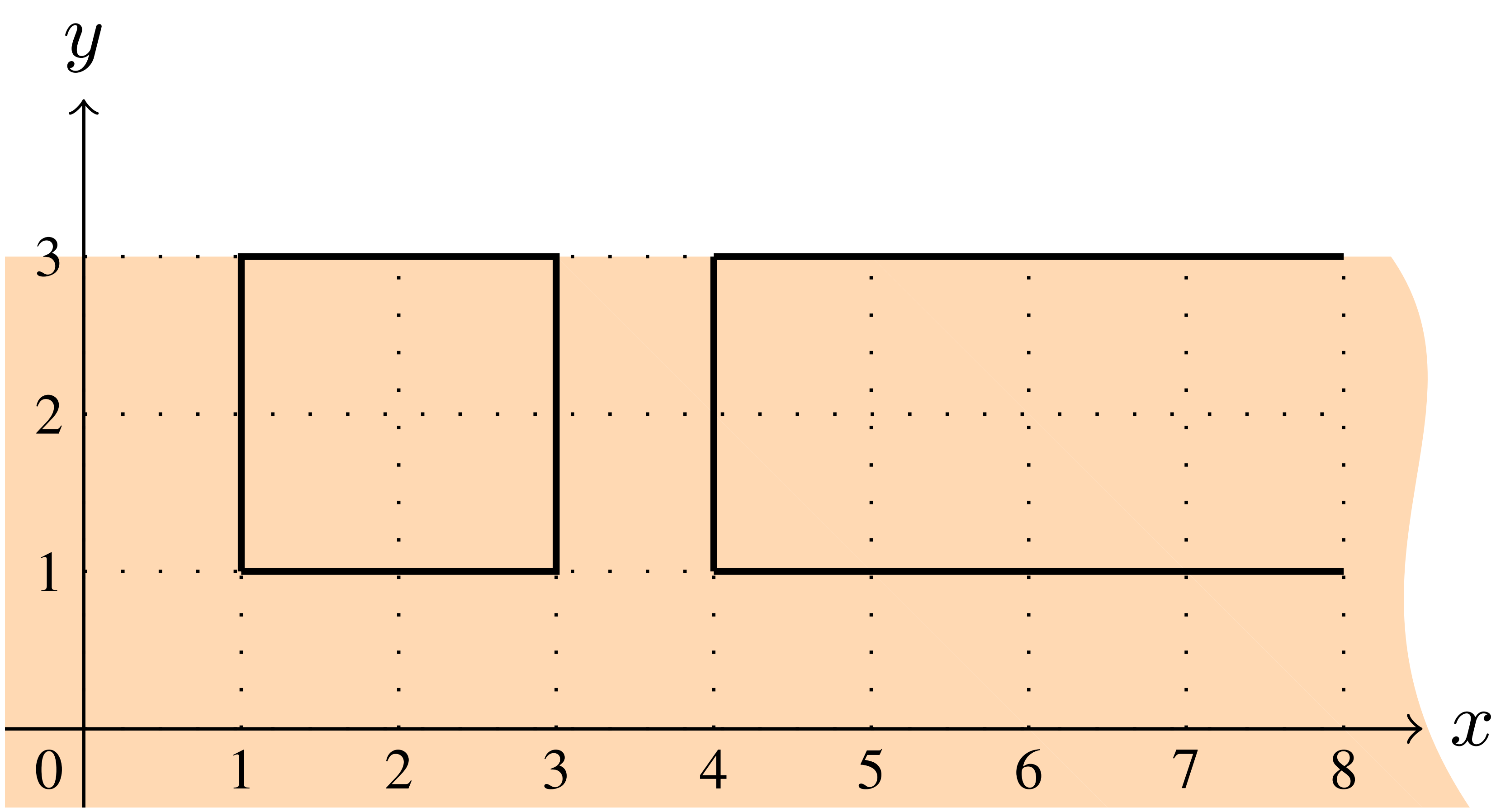
__Optimal Solution:__

$$y \leq 3 \wedge$$
$$x + y \leq \infty$$

Example

Objective functions: $\{y, x + y\}$

<u>Under-approximation</u> of optimal solution: $false$

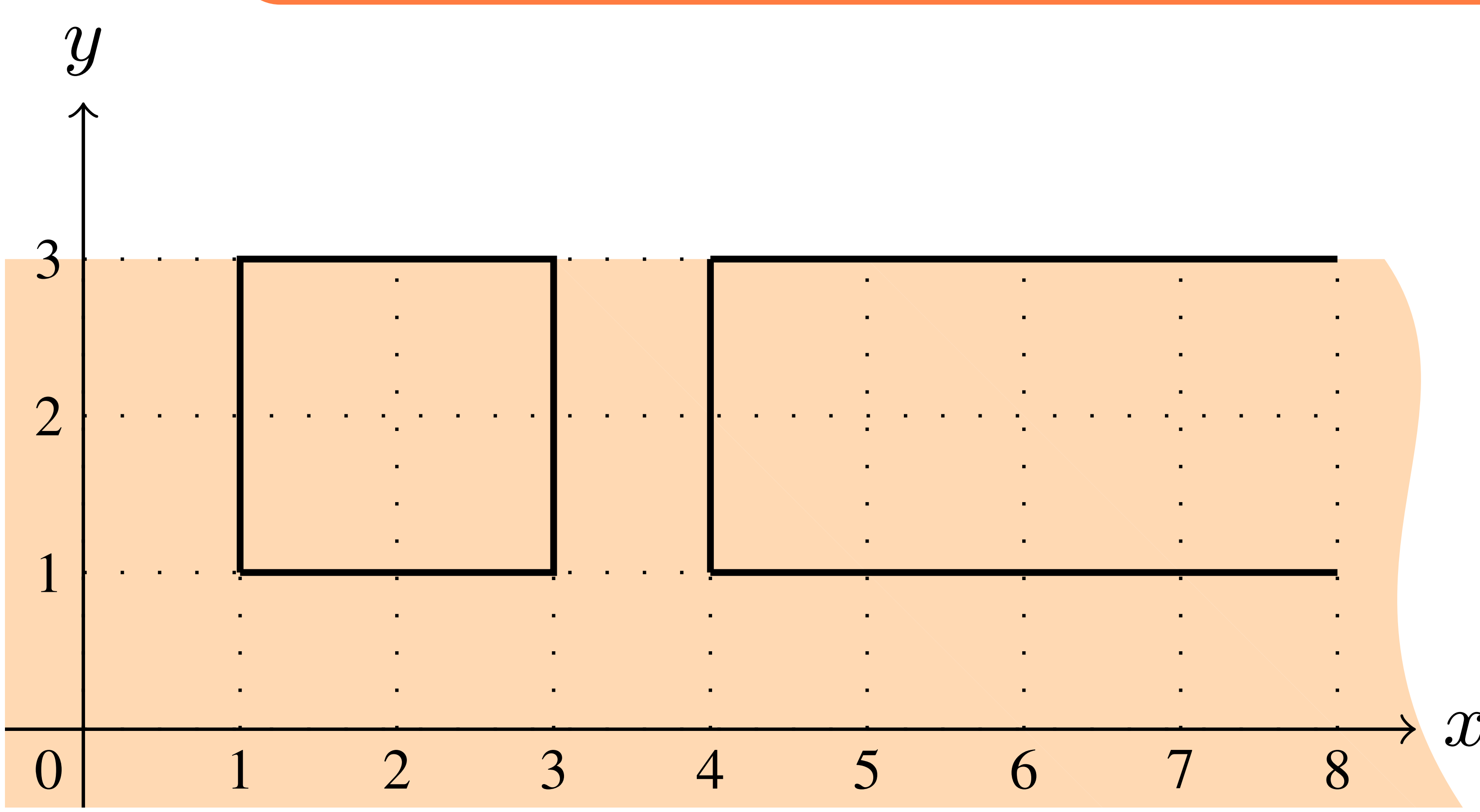


Example

Objective functions: $\{y, x + y\}$

<u>Under-approximation</u> of optimal solution: $false$

Phase 1: Grow under-approximation

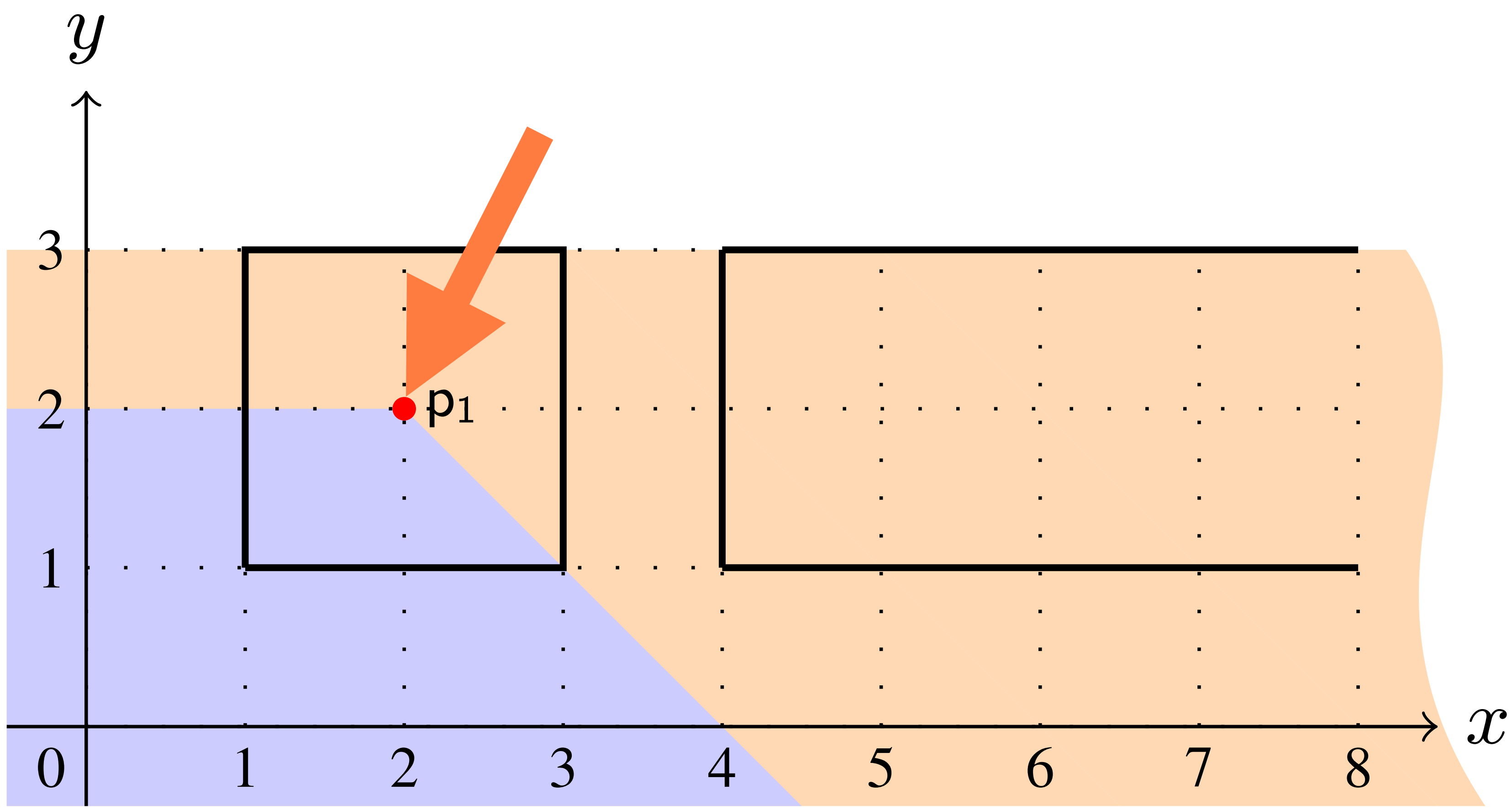


Example

Objective functions: $\{y, x + y\}$

<u>Under-approximation</u> of optimal solution: $false$

Phase 1: Grow under-approximation

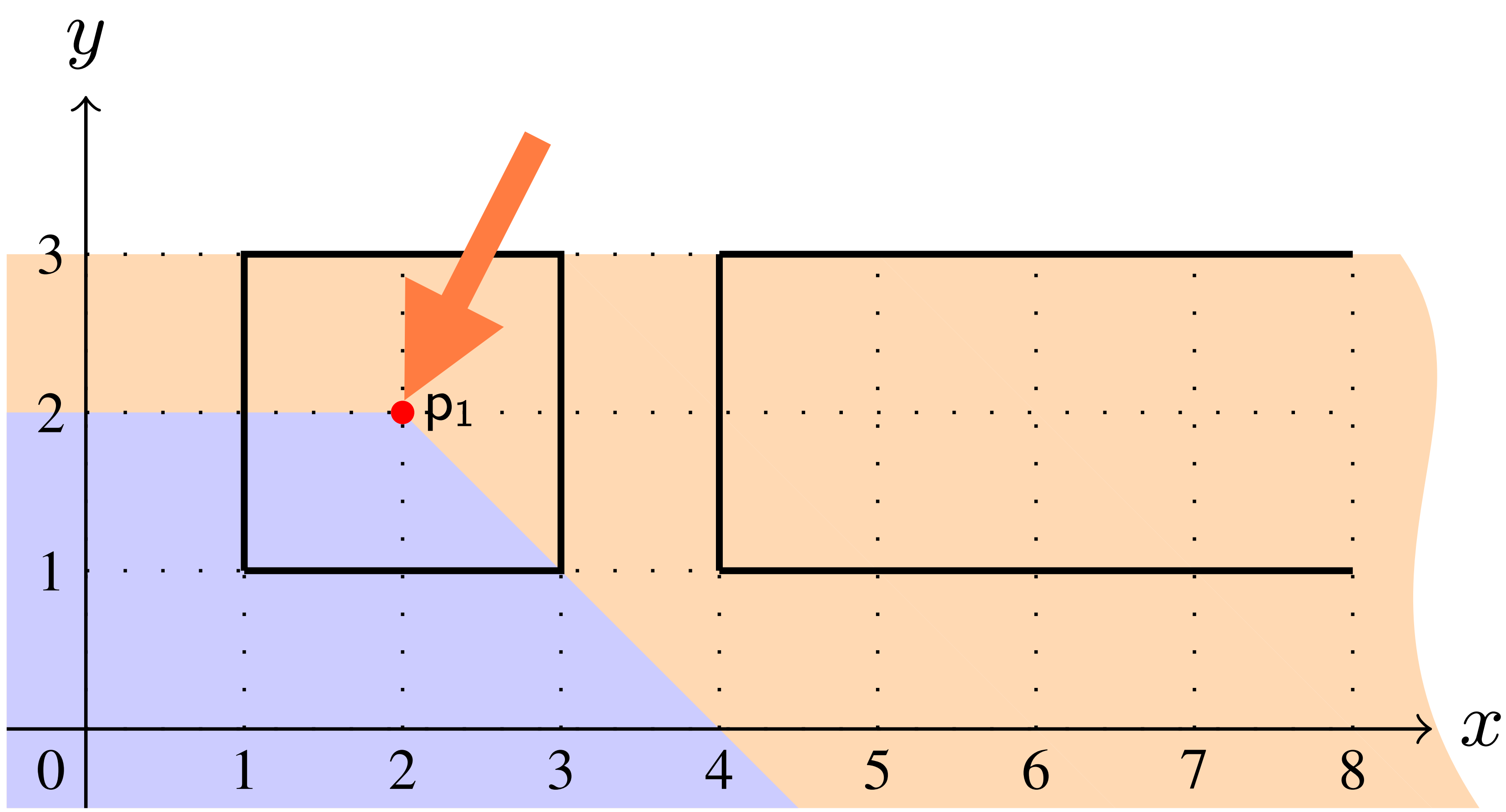


Example

Objective functions: $\{y, x + y\}$

Under-approximation of optimal solution: $false$

Phase 1: Grow under-approximation



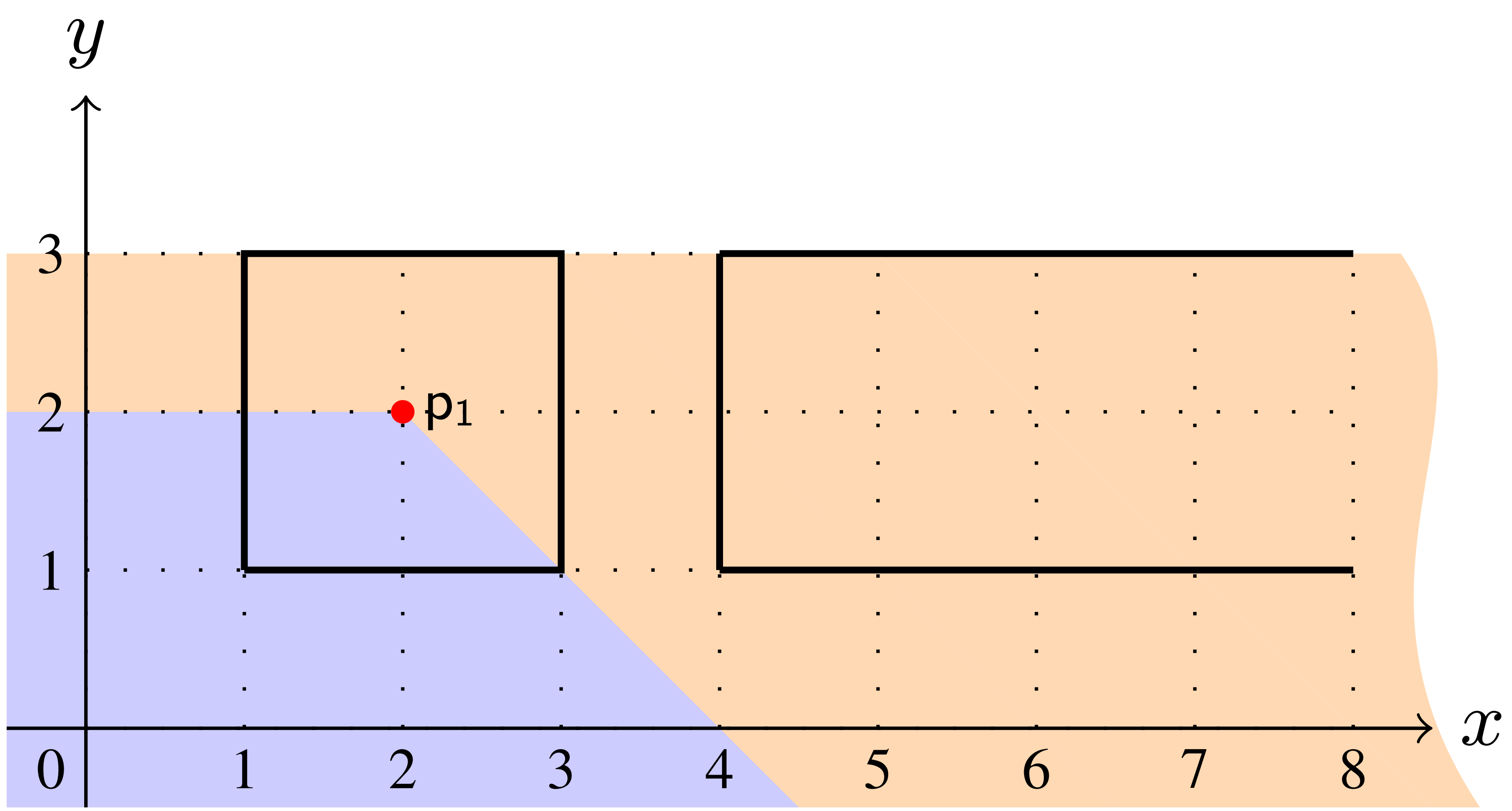
New under-approx:

$$y \le 2 \land$$
$$x + y \le 4$$

Example

Objective functions: $\{y, x + y\}$

Phase 2: Check if y *is unbounded*



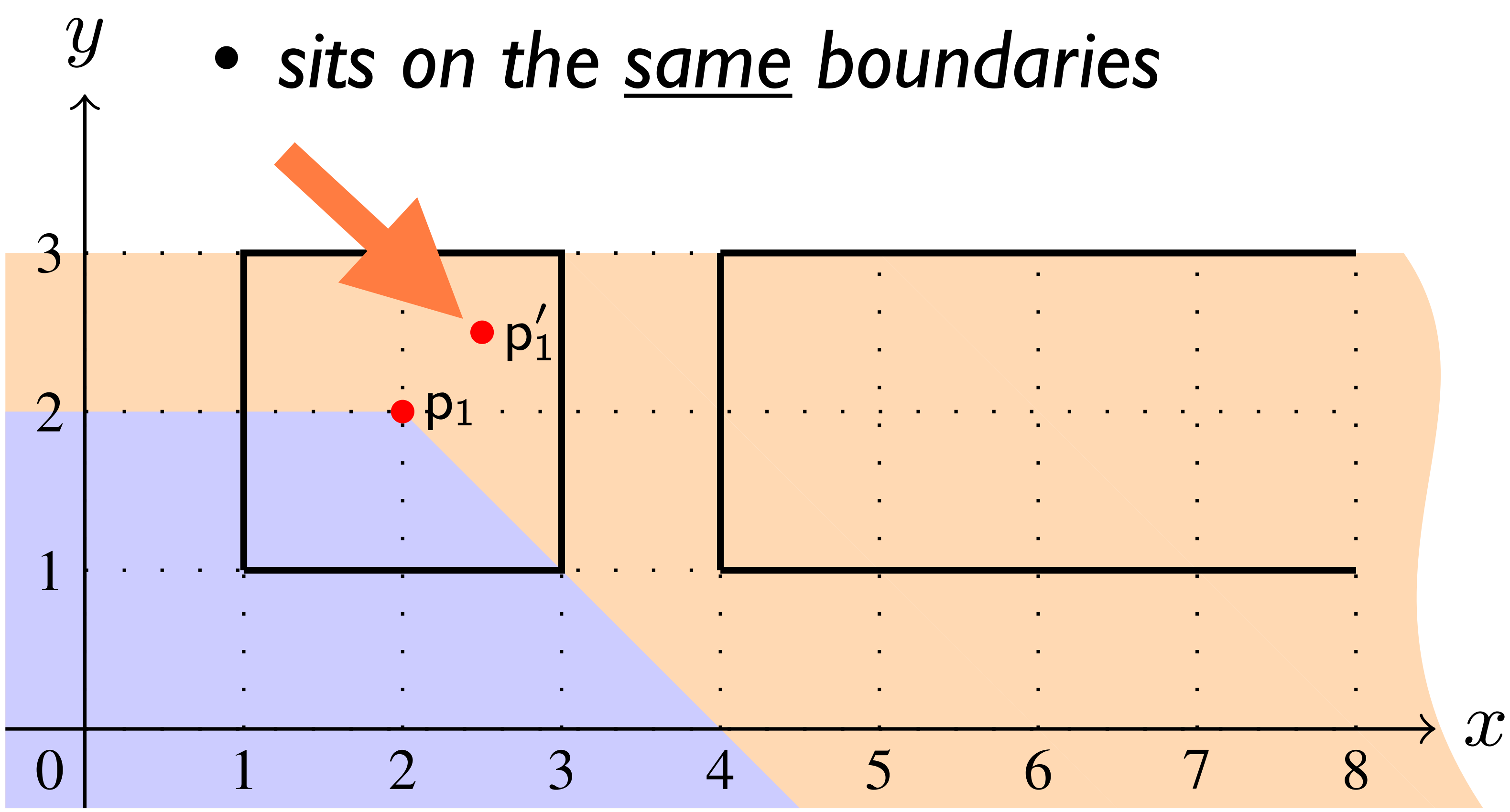
Example

Phase 2: Check if $\mathcal{Y}$ is unbounded

Pick point p_1

Find point p_1' s.t.

- *increases value of* $\mathcal{Y}$
- *sits on the <u>same</u> boundaries*



Example

Phase 2: Check if y is unbounded

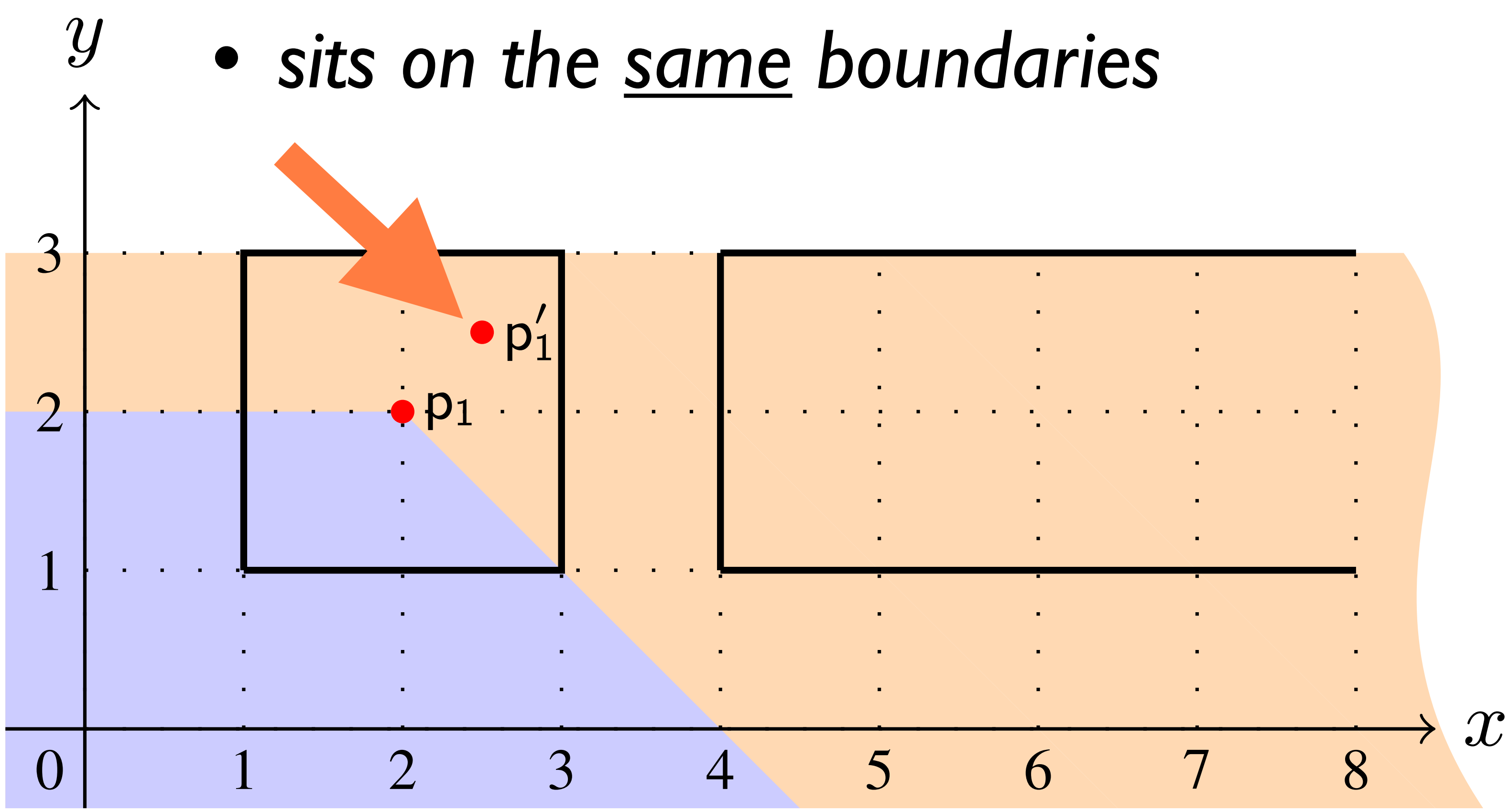
Pick point p_1

Find point p_1' s.t.

- *increases value of* y
- *sits on the <u>same</u> boundaries*

Find point p_2 s.t.

- *increases value of* y
- *sits on <u>more</u> boundaries*



Example

Phase 2: Check if y is unbounded

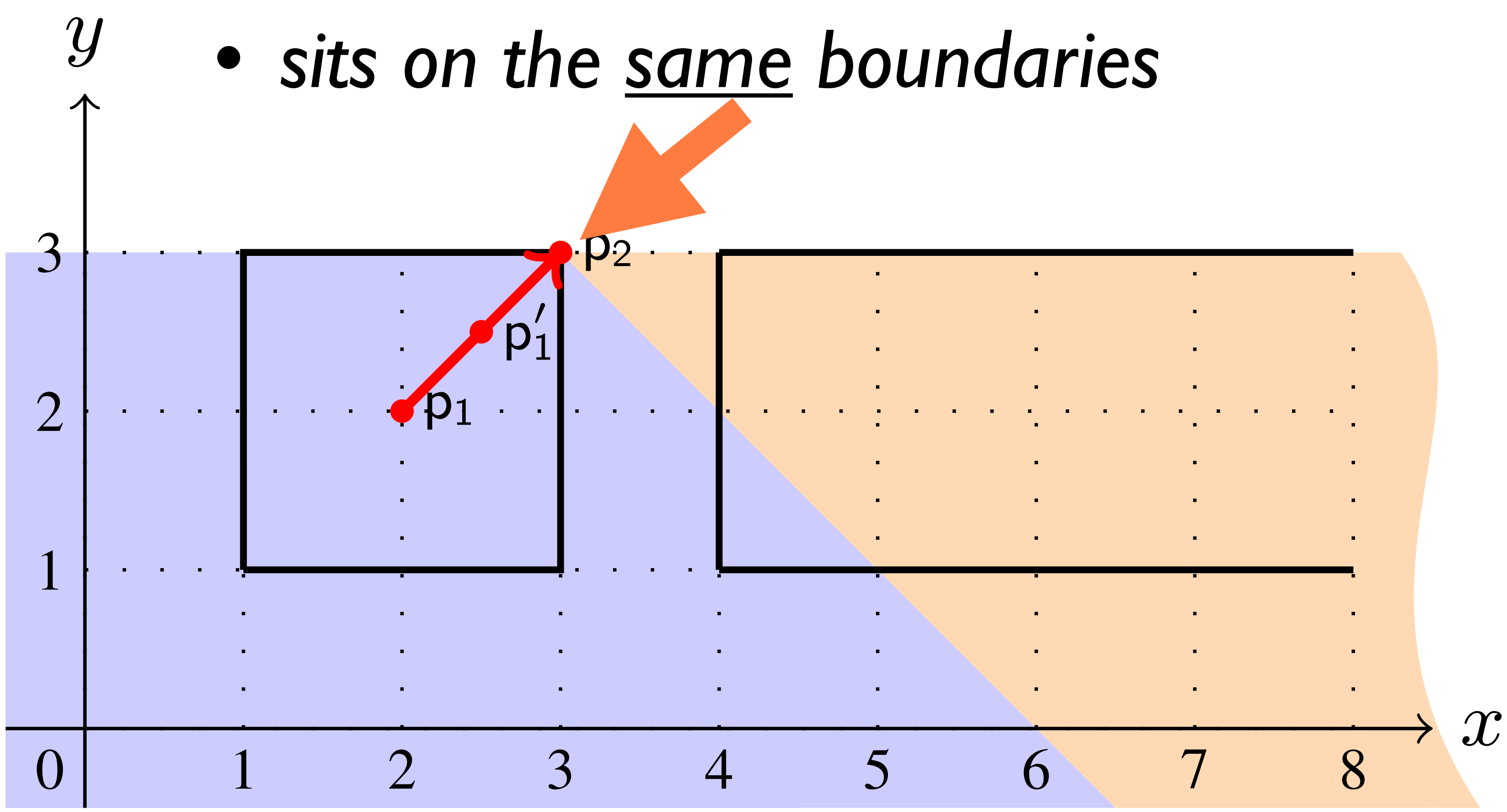
Pick point p_1

Find point p_1' s.t.

- *increases value of* y
- *sits on the <u>same</u> boundaries*

Find point p_2 s.t.

- *increases value of* y
- *sits on <u>more</u> boundaries*



Example

Phase 2: Check if y is unbounded

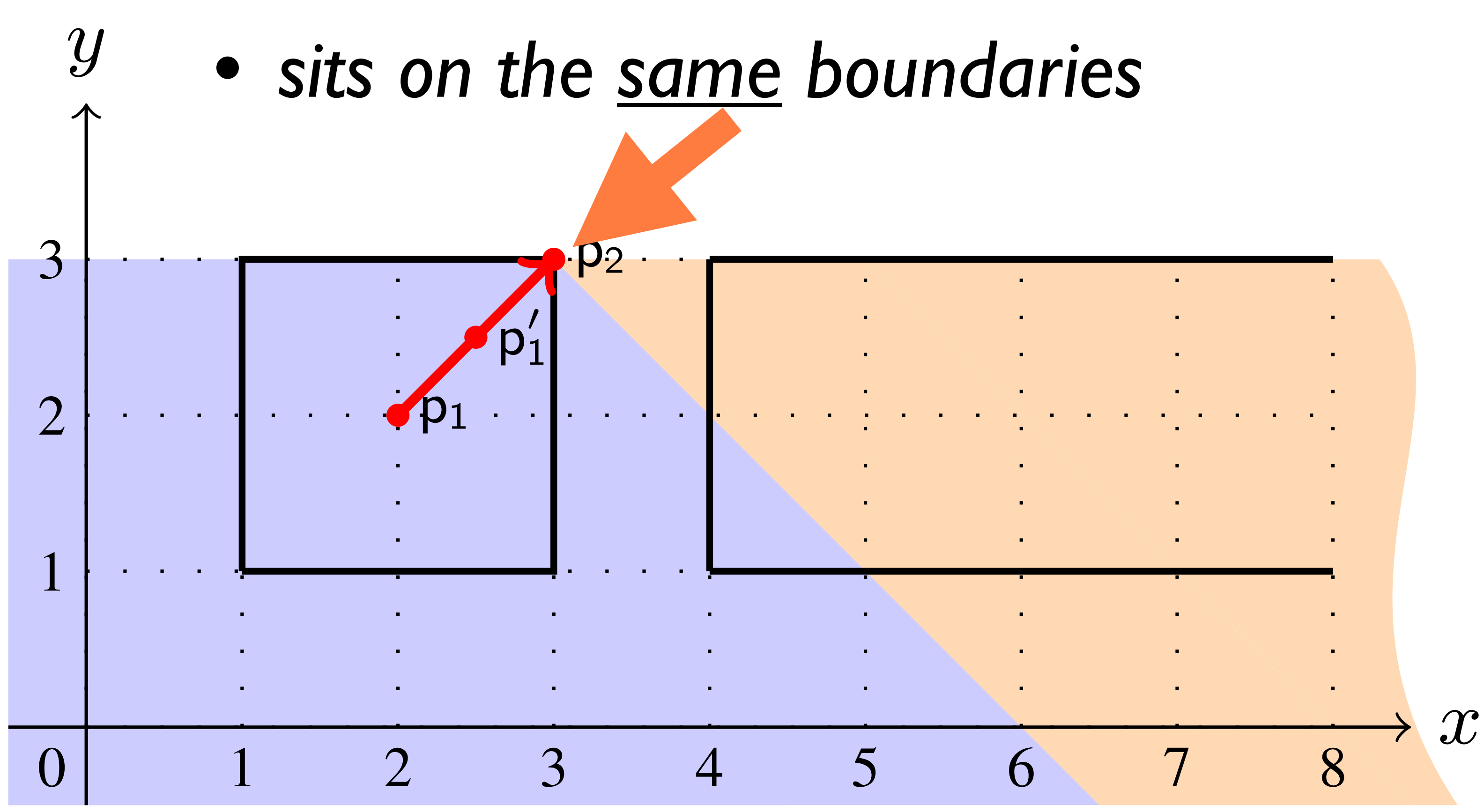
Pick point p_1

Find point p_1' s.t.

- *increases value of y*
- *sits on the same boundaries*

Find point p_2 s.t.

- *increases value of y*
- *sits on more boundaries*



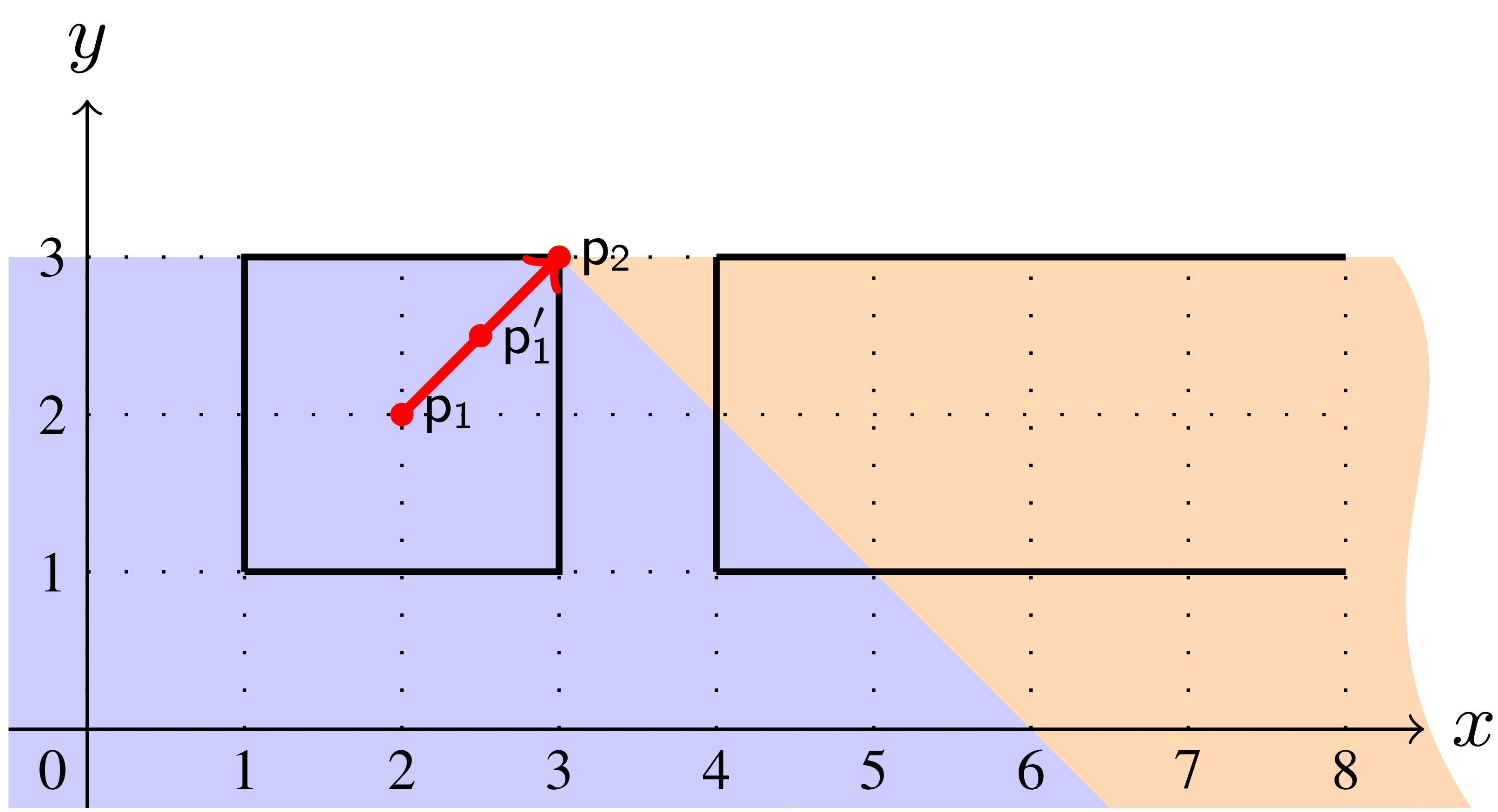
New under-approx:

$$y \leq 3 \wedge$$
$$x + y \leq 6$$

Example

Objective functions: $\{y, x + y\}$

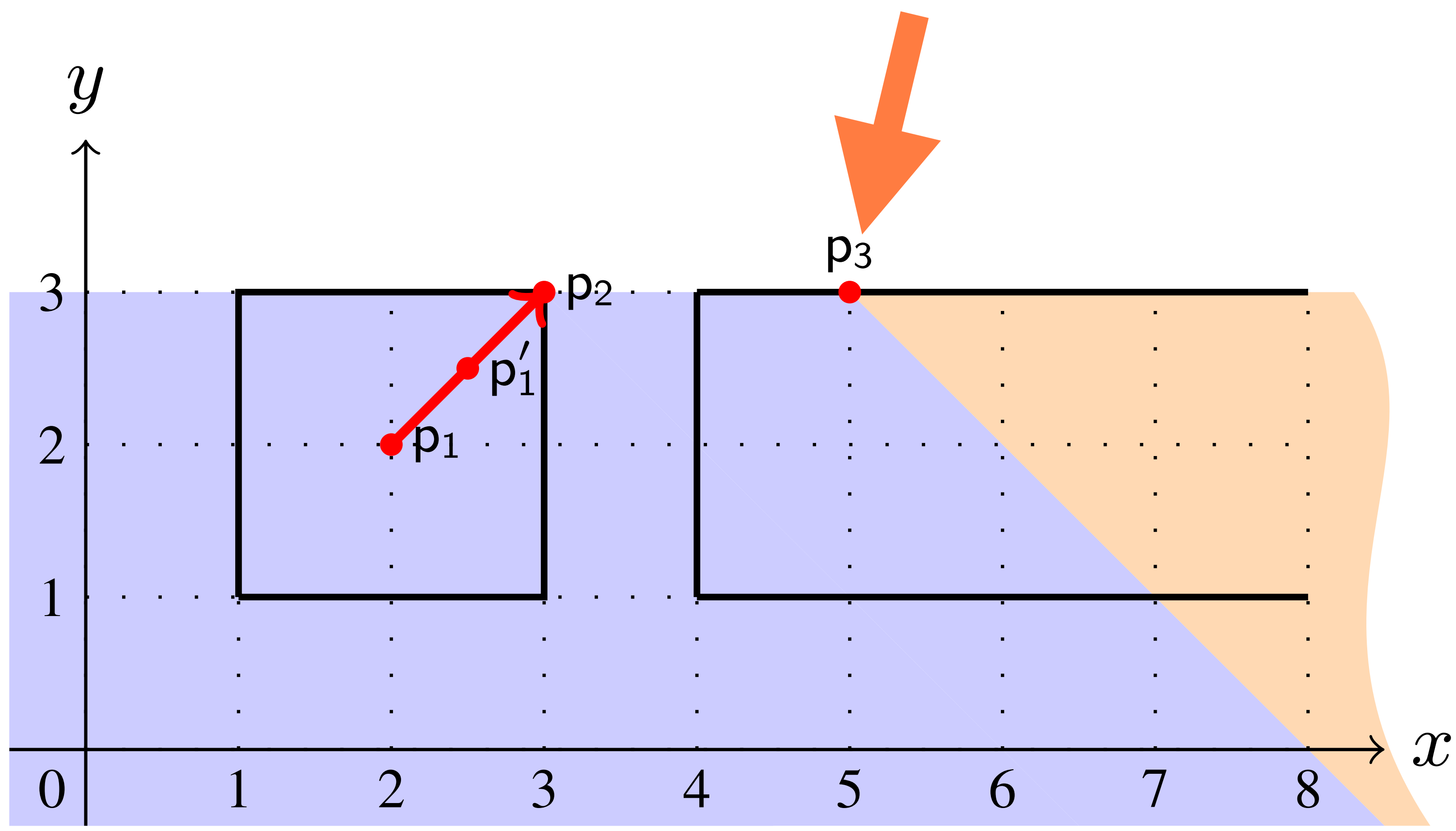
Phase 1: Grow under-approximation



Example

Objective functions: $\{y, x + y\}$

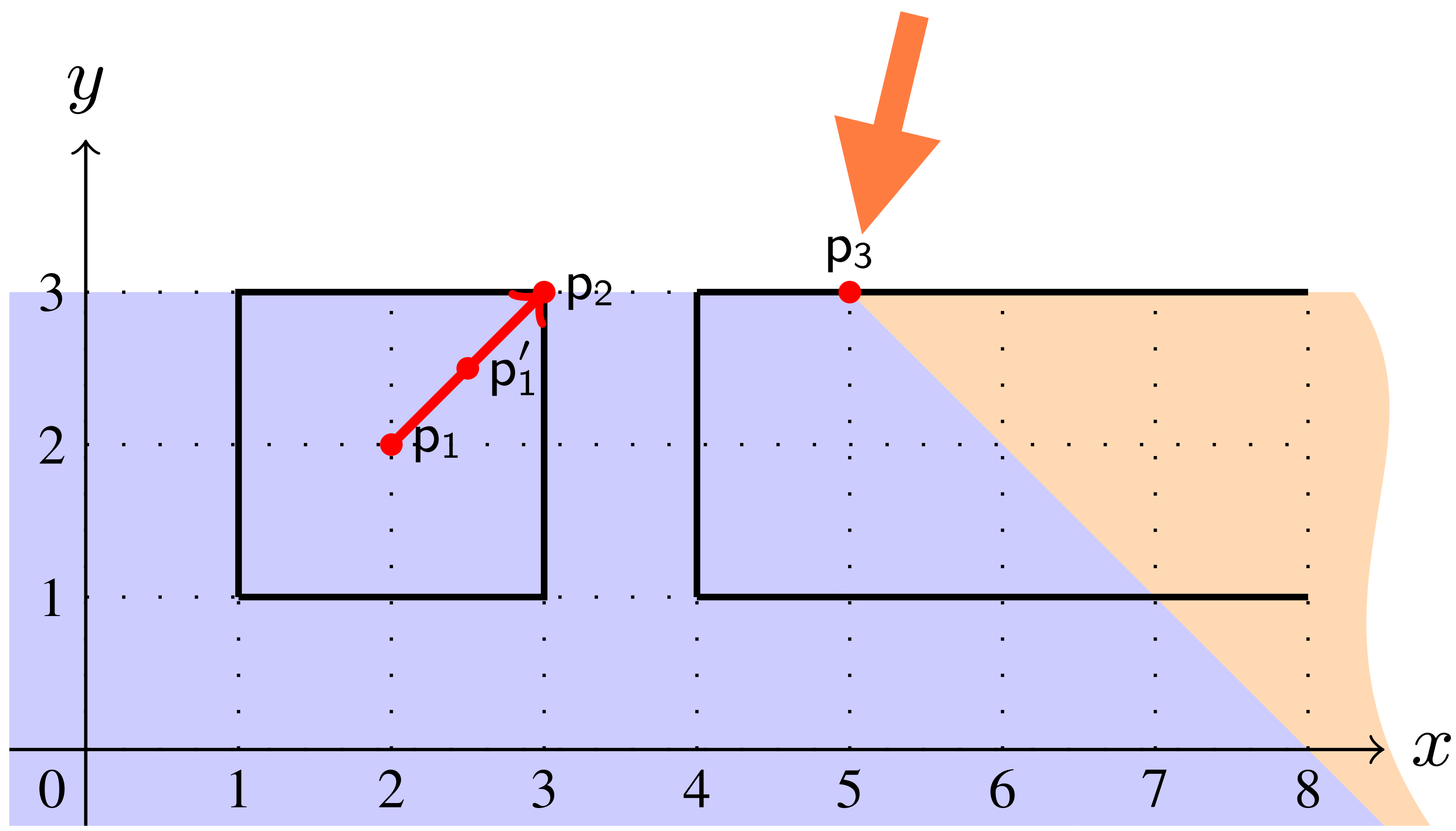
Phase 1: Grow under-approximation



Example

Objective functions: $\{y, x+y\}$

Phase 1: Grow under-approximation



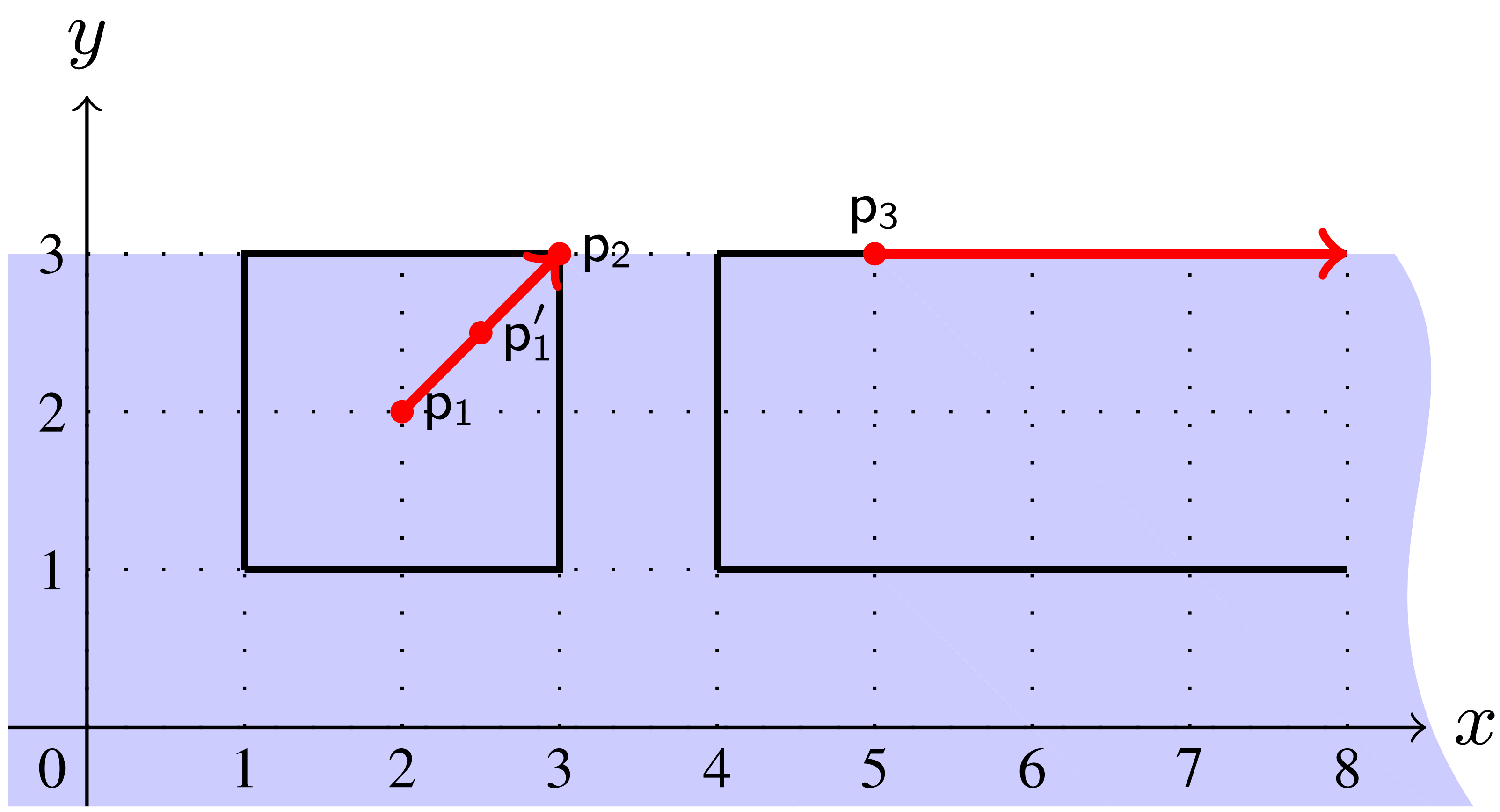
New under-approx:

$$y \le 3 \wedge$$
$$x + y \le 8$$

Example

Objective functions: $\{y, x + y\}$

Phase 2: Check if $x + y$ *is unbounded*

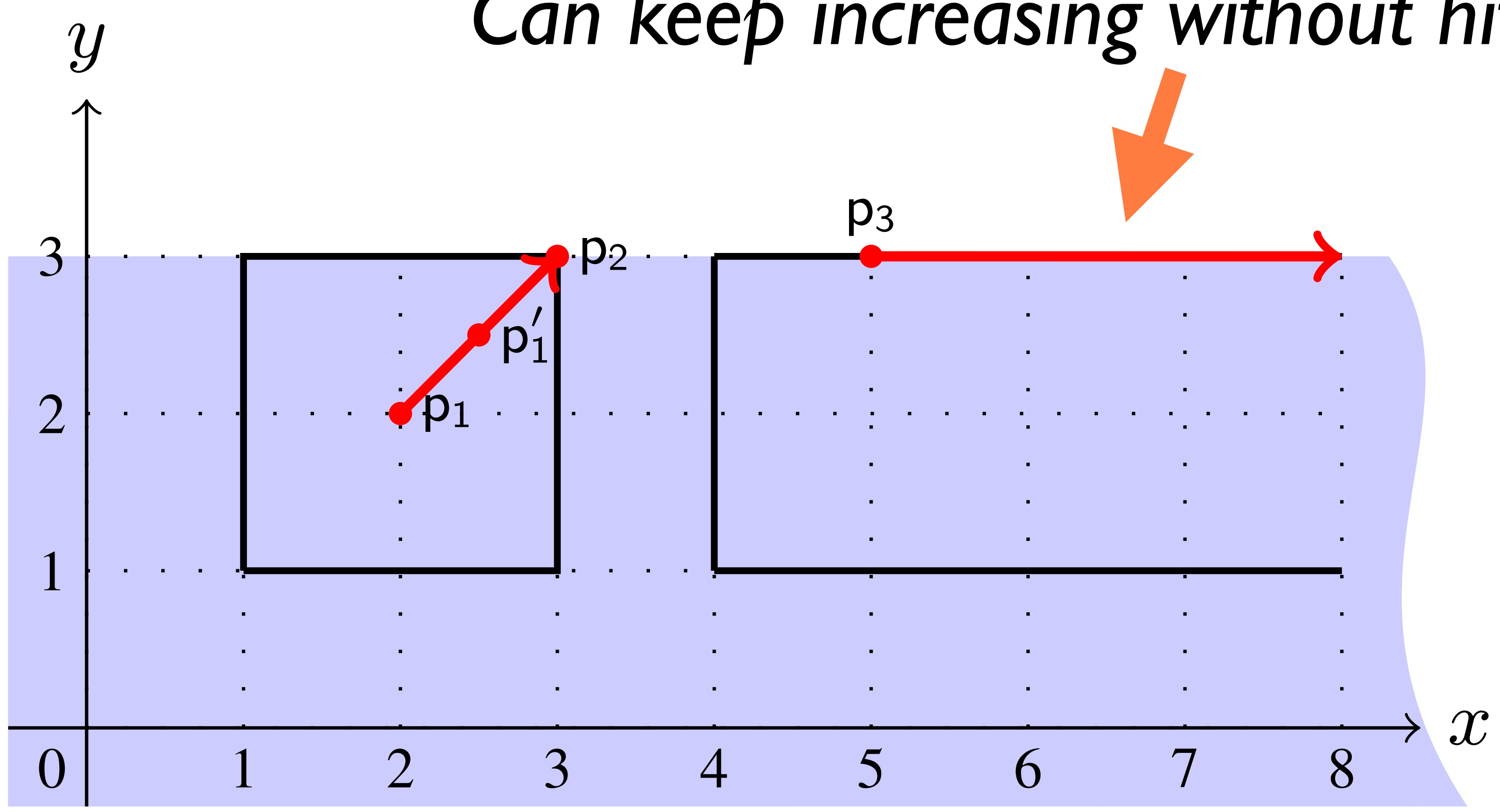


Example

Objective functions: $\{y, x+y\}$

Phase 2: Check if $x+y$ *is unbounded*

Can keep increasing without hitting a boundary

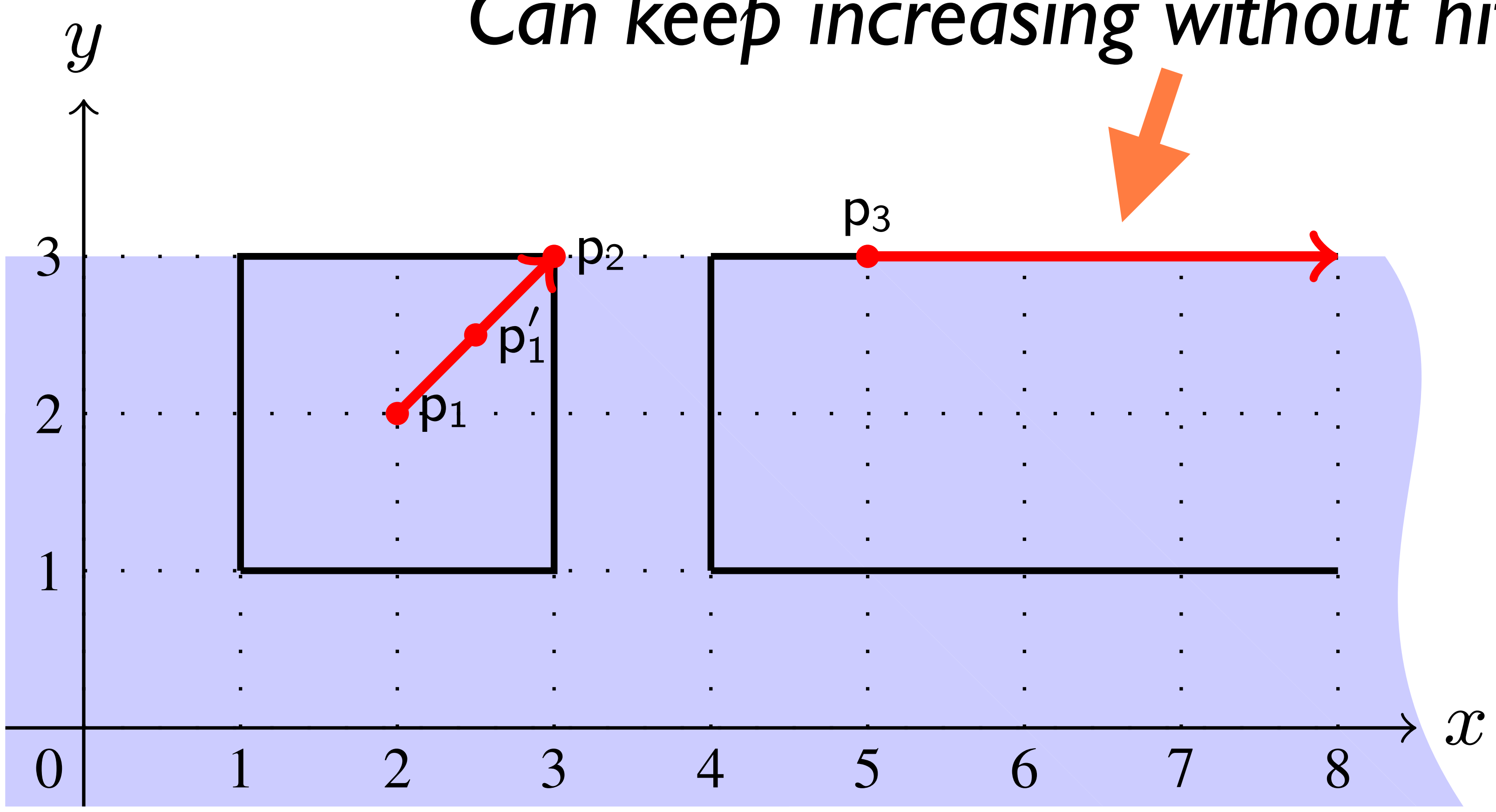


Example

Objective functions: $\{y, x + y\}$

Phase 2: Check if $x + y$ *is unbounded*

Can keep increasing without hitting a boundary



__*Optimal solution:*__

$$y \leq 3 \wedge$$
$$x + y \leq \infty$$

Symba in a Nutshell

Alternate between two phases

- *Sampling: grow under-approximation*
- *Check if objective function is unbounded*

Symba in a Nutshell

Alternate between two phases

- *Sampling: grow under-approximation*
- *Check if objective function is unbounded*

Fair alternation ensures completeness

Symba in a Nutshell

Alternate between two phases

- *Sampling: grow under-approximation*
- *Check if objective function is unbounded*

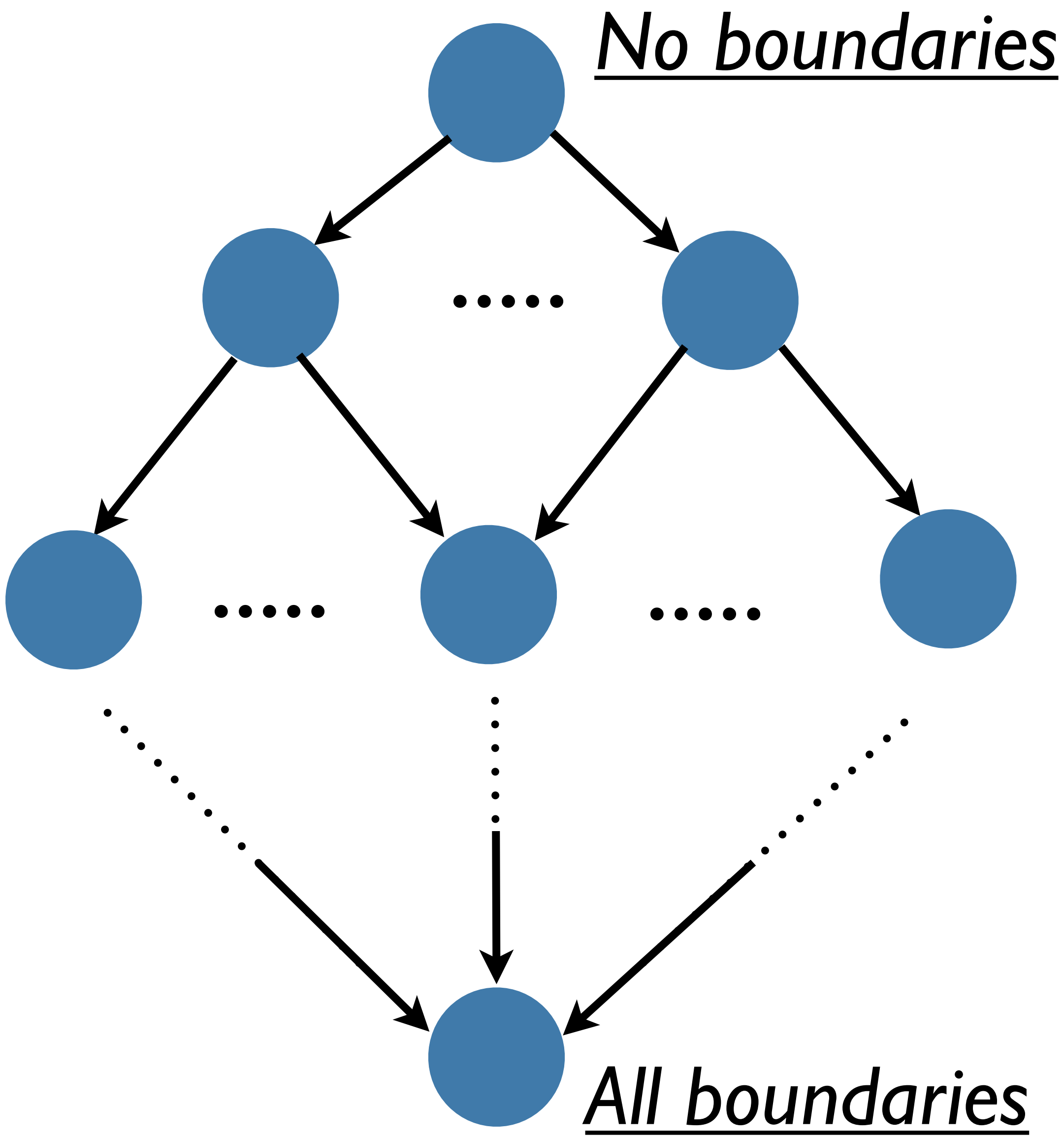
<u>*Fair*</u> alternation ensures completeness

Algorithm also maintains an <u>*over-approx*</u>

- *See paper*

Symba Abstractly

Arrange infinitely many models into finitely many boundary classes



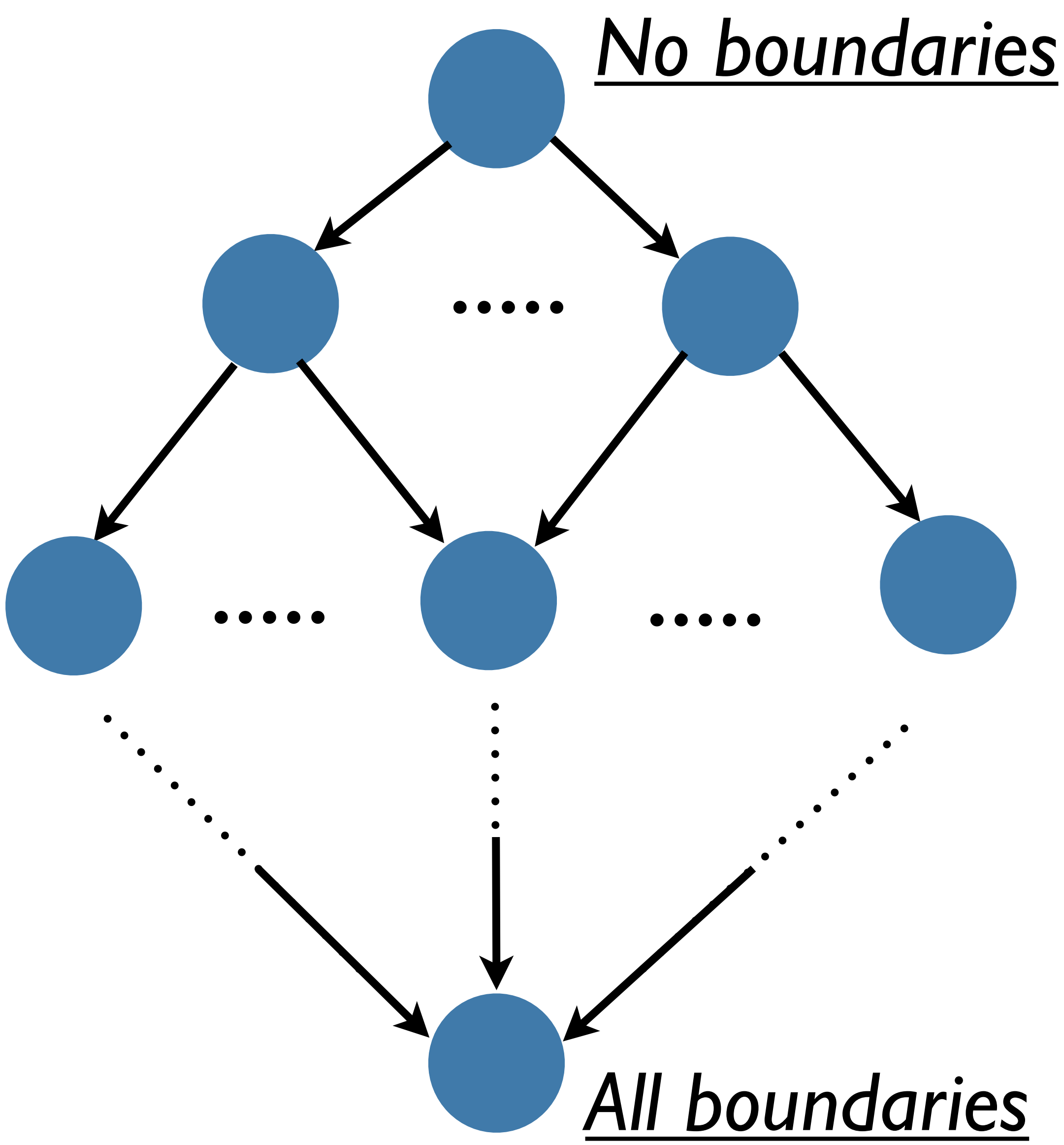
Symba Abstractly

Arrange infinitely many models into finitely many boundary classes

Find p_1, p_2 in same boundary class s.t.

$$f(p_1) < f(p_2)$$

no p_3 exists in stronger boundary class where $f(p_3) \geq f(p_2)$



Symba Abstractly

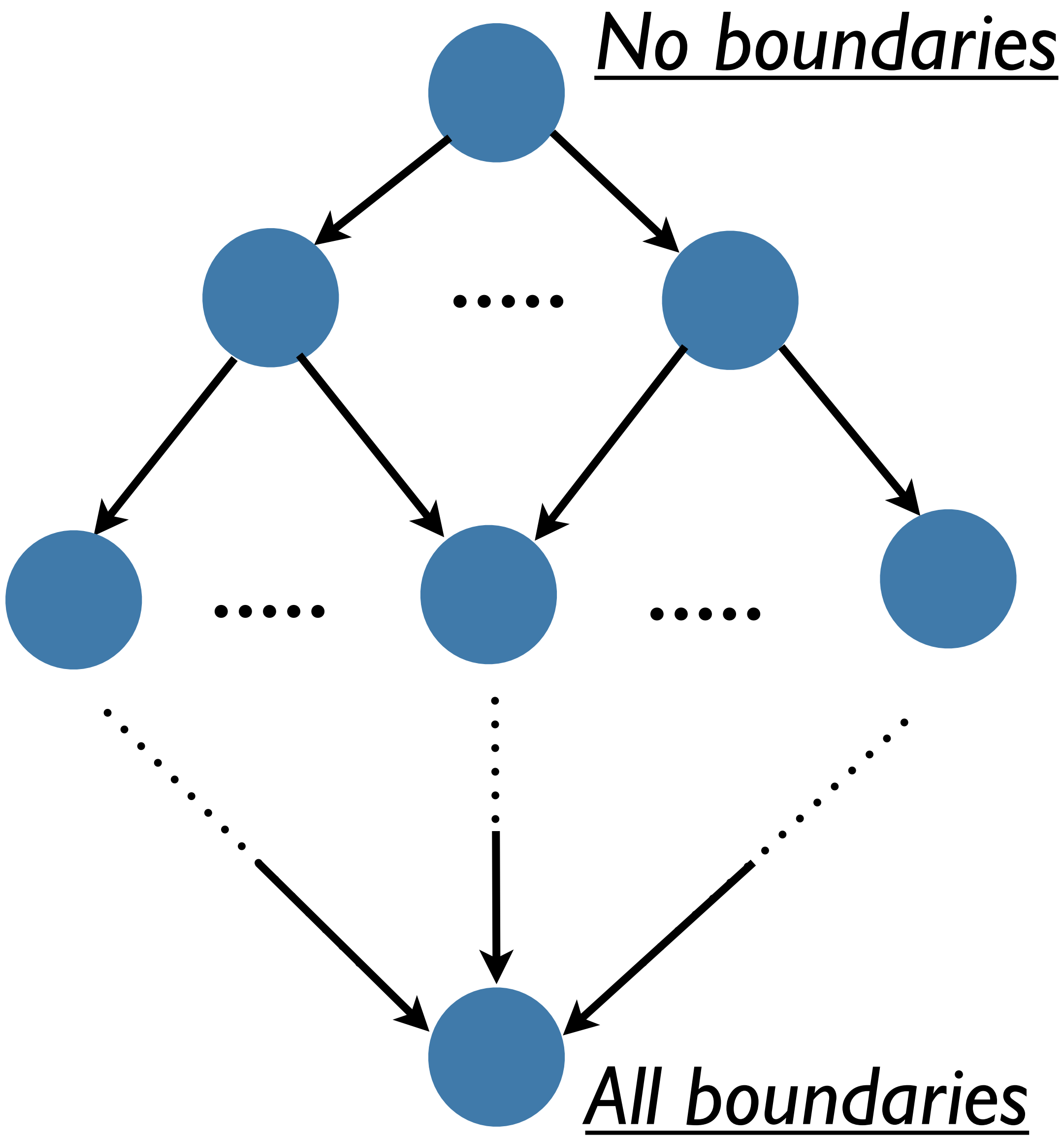
Arrange infinitely many models into finitely many boundary classes

Find p_1, p_2 in same boundary class s.t.

$f(p_1) < f(p_2)$

no p_3 exists in stronger boundary class where $f(p_3) \geq f(p_2)$

Necessary and sufficient condition to prove unboundedness of f



Symba Abstractly

Arrange infinitely many models into finitely many boundary classes

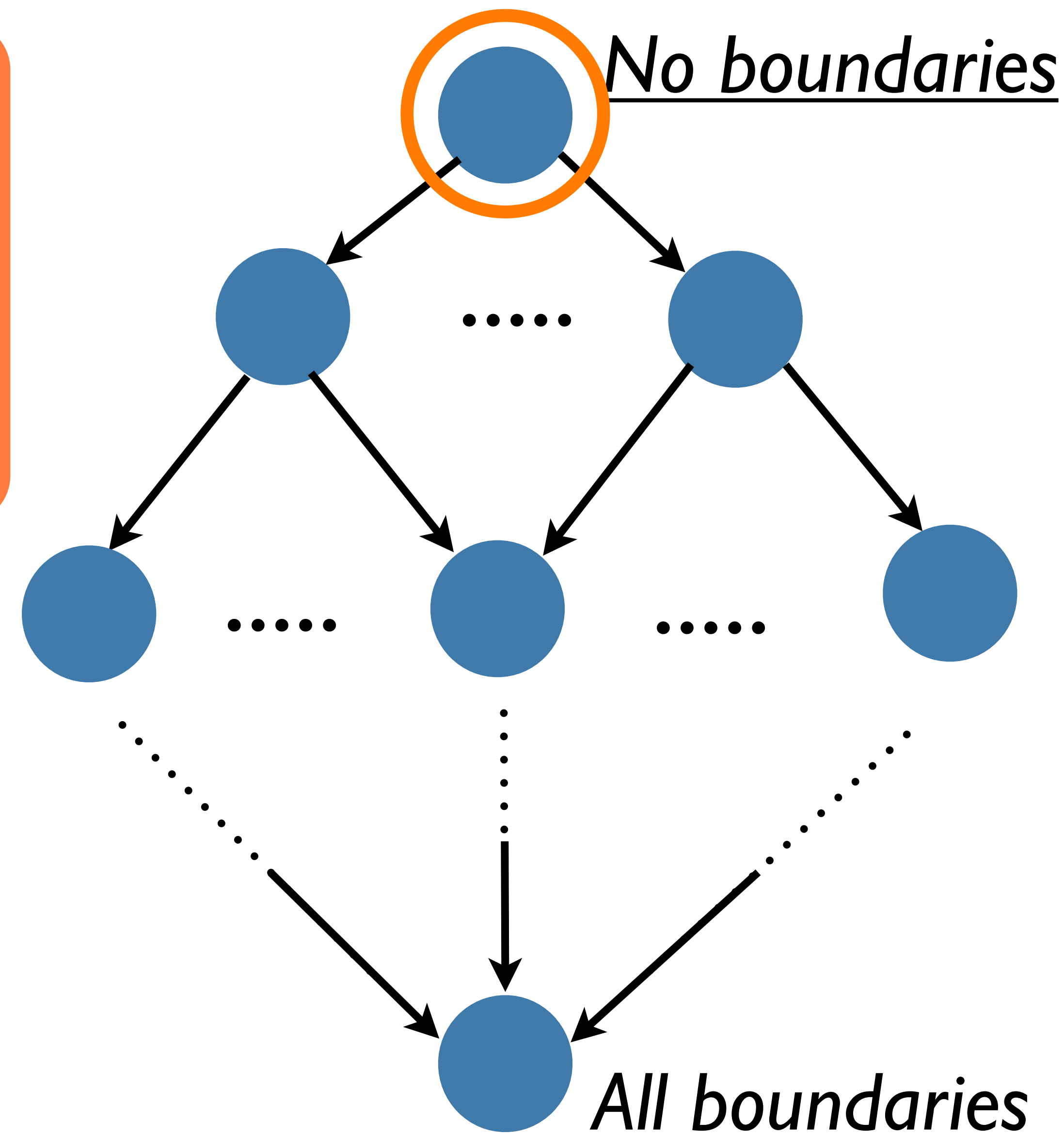
Find p_1, p_2 in same boundary class s.t.

$$f(p_1) < f(p_2)$$

no p_3 exists in stronger boundary class where $f(p_3) \geq f(p_2)$

Necessary and sufficient condition to prove unboundedness of f

Symba searches through lattice of classes!



Symba Abstractly

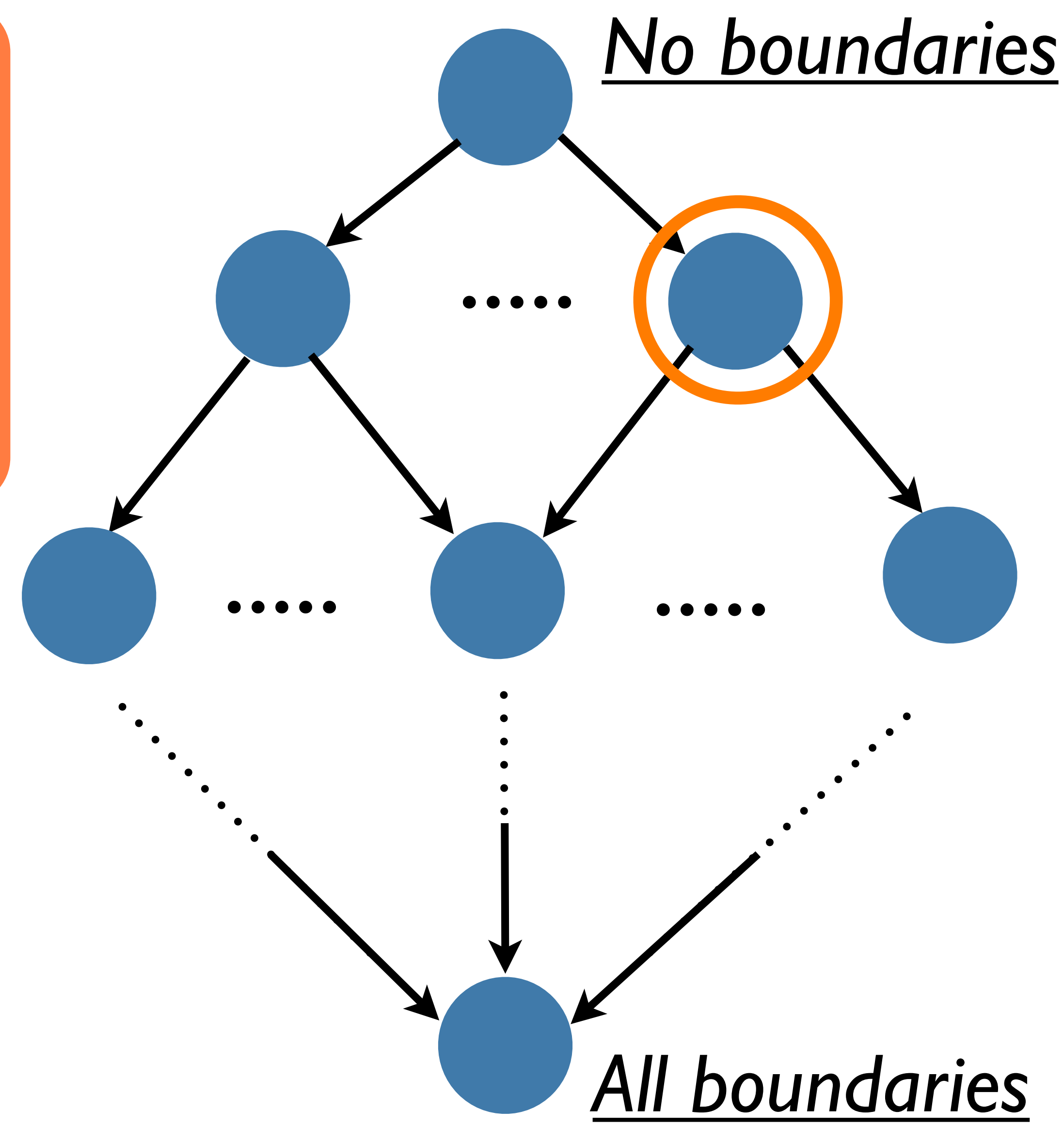
Arrange infinitely many models into finitely many boundary classes

Find p_1, p_2 in same boundary class s.t.

$$f(p_1) < f(p_2)$$

no p_3 exists in stronger boundary class where $f(p_3) \geq f(p_2)$

Necessary and sufficient condition to prove unboundedness of f



Symba searches through lattice of classes!

Symba Abstractly

Arrange infinitely many models into finitely many ***boundary classes***

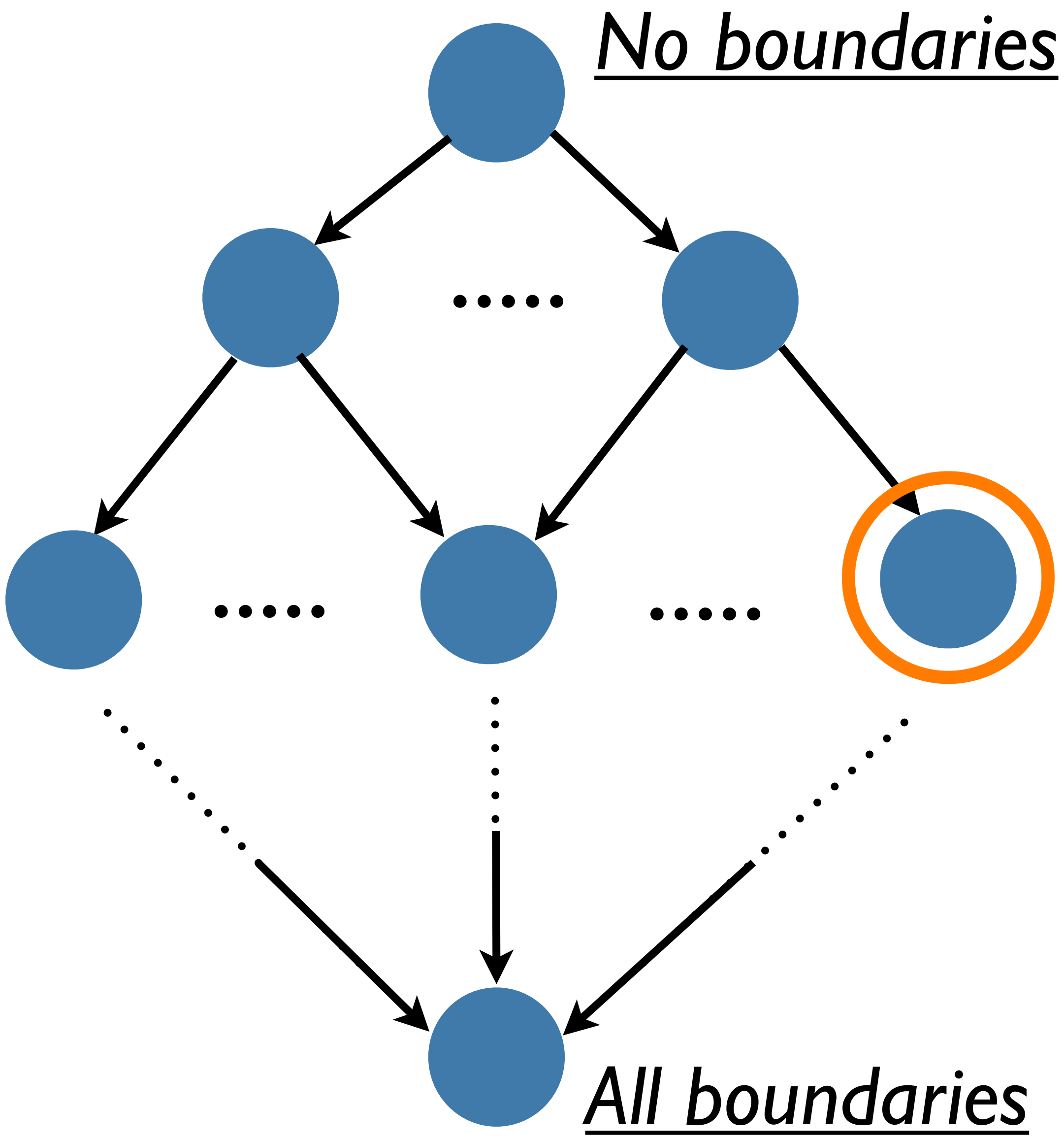
Find p_1, p_2 in same boundary class s.t.

$$f(p_1) < f(p_2)$$

no p_3 exists in stronger boundary class where $f(p_3) \geq f(p_2)$

Necessary and sufficient condition to prove unboundedness of f

Symba searches through lattice of classes!



Symba Abstractly

Arrange infinitely many models into finitely many boundary classes

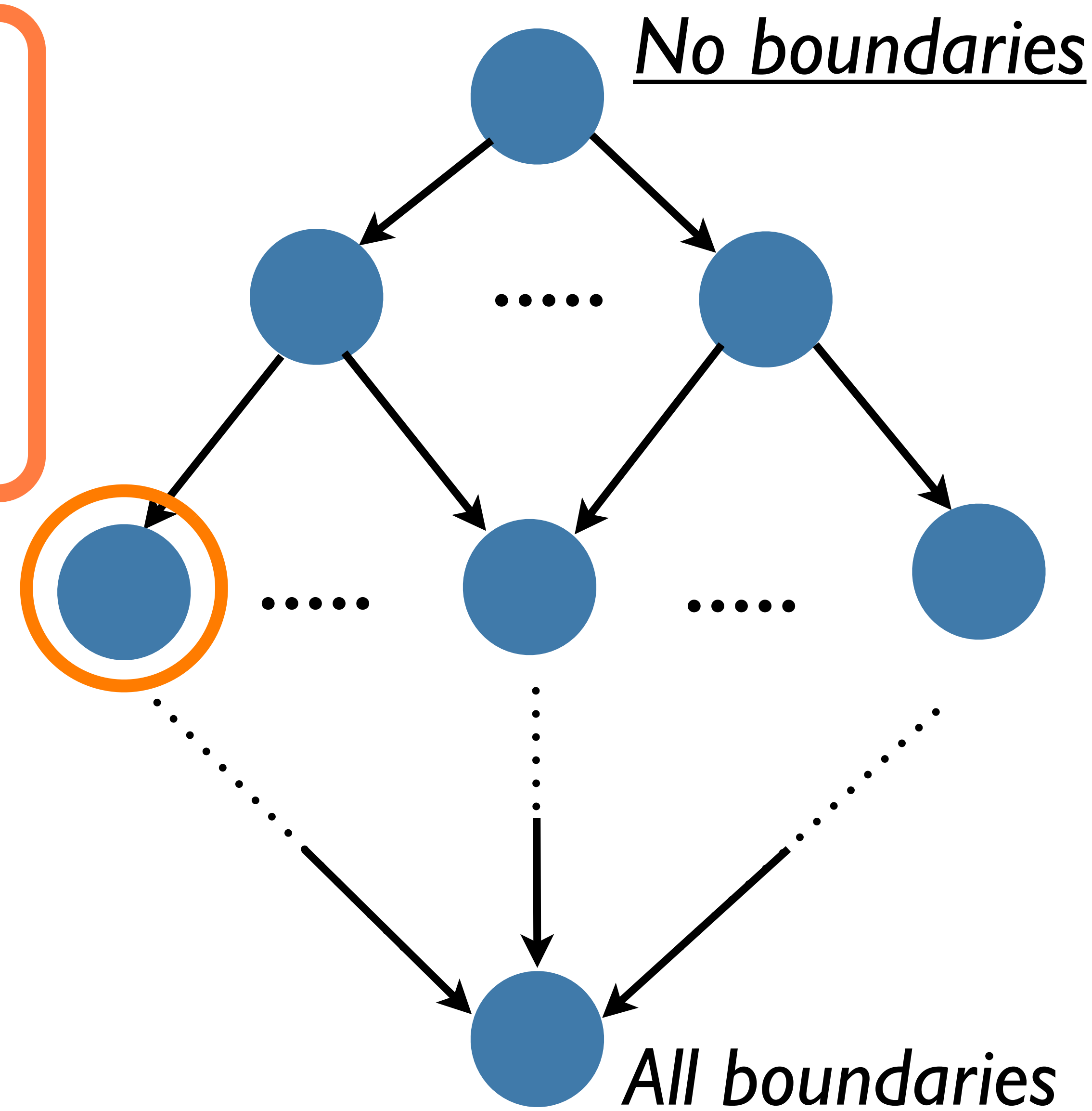
Find p_1, p_2 in same boundary class s.t.

$$f(p_1) < f(p_2)$$

no p_3 exists in stronger boundary class where $f(p_3) \geq f(p_2)$

Necessary and sufficient condition to prove unboundedness of f

Symba searches through lattice of classes!



Application

Implemented Symba using Z3

Application: Computing *precise abstract transformers*:

- *TCM domains (intervals, octagons, etc.)*
  [Sankaranarayanan et al., VMCAI‘05]
- *Complex transition relations (multiple paths)*

Application

Implemented Symba using Z3

Application: Computing *precise abstract transformers*:

- *TCM domains (intervals, octagons, etc.)* [Sankaranarayanan et al., VMCAI'05]
- *Complex transition relations (multiple paths)*

S

Initial states

Application

Implemented Symba using Z3

Application: Computing *precise abstract transformers*:

- *TCM domains (intervals, octagons, etc.)* [Sankaranarayanan et al., VMCAI'05]
- *Complex transition relations (multiple paths)*

$S \qquad T$

Initial states

Transition relation (multiple paths)

Application

Implemented Symba using Z3

Application: Computing *precise abstract transformers*:

- *TCM domains (intervals, octagons, etc.)* [Sankaranarayanan et al., VMCAI'05]
- *Complex transition relations (multiple paths)*

$$S \wedge T$$

Initial states

Transition relation (multiple paths)

Application

Implemented Symba using Z3

Application: Computing *precise abstract transformers*:

- *TCM domains (intervals, octagons, etc.)* [Sankaranarayanan et al., VMCAI'05]
- *Complex transition relations (multiple paths)*

$$S \wedge T$$

Initial states

Transition relation (multiple paths)

Objective functions:

Intervals domain:

$$\{x, -x, \ldots\}$$

Application

Implemented Symba using Z3

Application: Computing *precise abstract transformers*:

- *TCM domains (intervals, octagons, etc.)* [Sankaranarayanan et al., VMCAI'05]
- *Complex transition relations (multiple paths)*

$$S \wedge T$$

Initial states

Transition relation (multiple paths)

Objective functions:

Intervals domain:

$$\{x, -x, \ldots\}$$

Octagons domain:

$$\{x + y, \ldots\}$$

Evaluation

Instrumented UFO [CAV'12] to generate abstract post queries from SV-COMP programs

- *Took the ~1000 hardest benchmarks*
- *Average # of variables: ~900 (max: ~19,000)*
- *Average # of objective functions: 56 (max: 386)*

Evaluation

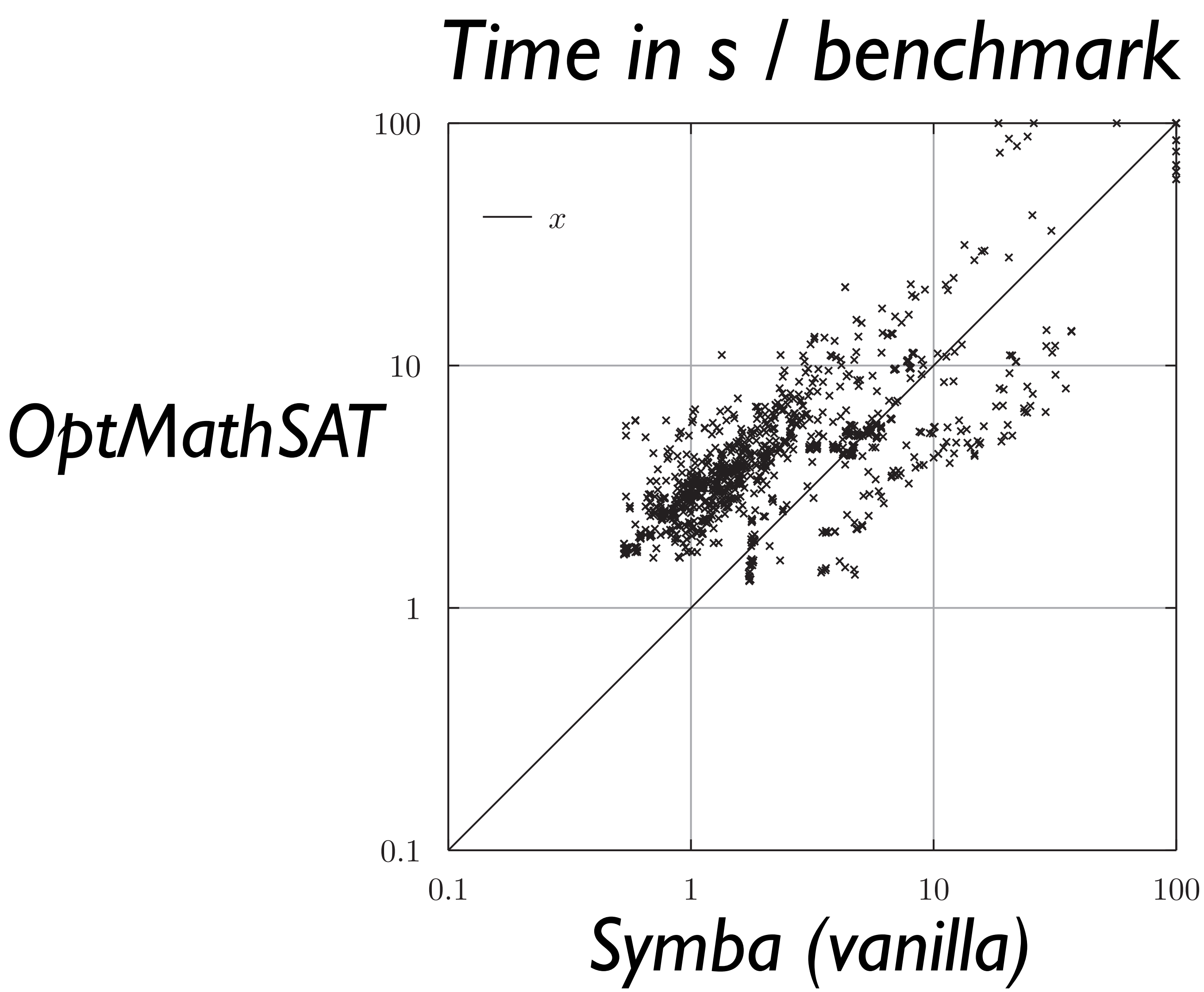
Compared w/ OptMathSAT [Sebastiani & Tomasi IJCAR'12]

- *Modifies SIMPLEX within SMT solver to find a local optimum*
- *Handles a single objective function at a time*

Evaluation

Compared w/ OptMathSAT [Sebastiani & Tomasi IJCAR'12]

- *Modifies SIMPLEX within SMT solver to find a local optimum*
- *Handles a single objective function at a time*



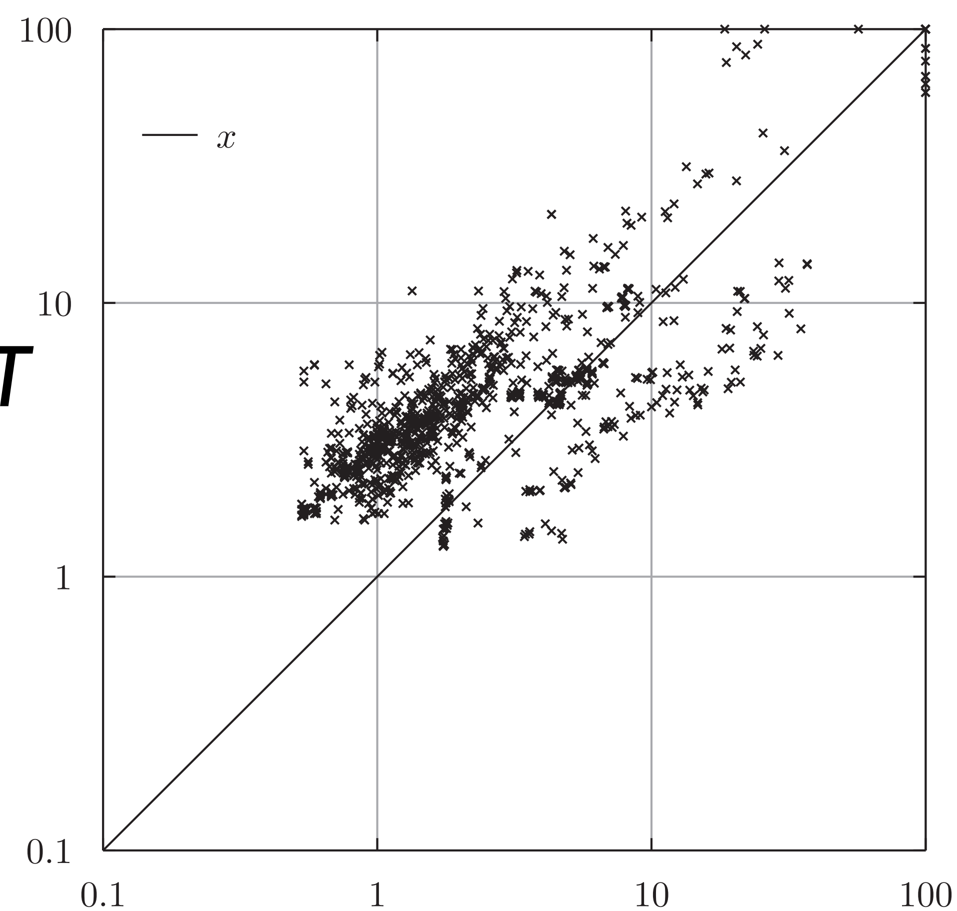
Evaluation

Compared w/ OptMathSAT [Sebastiani & Tomasi IJCAR'12]

- *Modifies SIMPLEX within SMT solver to find a local optimum*
- *Handles a single objective function at a time*

Time in s / benchmark

OptMathSAT



Symba (vanilla)

Symba outperforms OptMathSAT

<u>Average speedup: 2.2x</u>

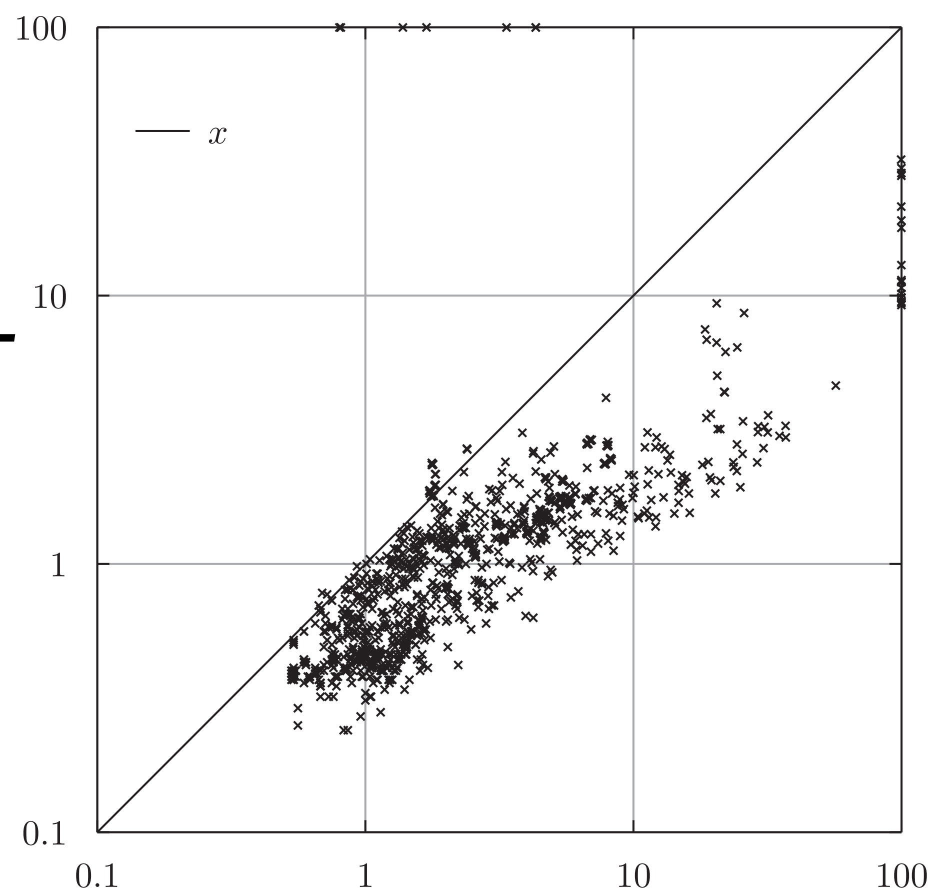
Evaluation

Implemented [Sebastiani & Tomasi, IJCAR'12] within Z3

- *Extended it to optimize multiple objectives simultaneously*

Time in s / benchmark

Our OptMathSAT (in Z3)



Symba (vanilla)

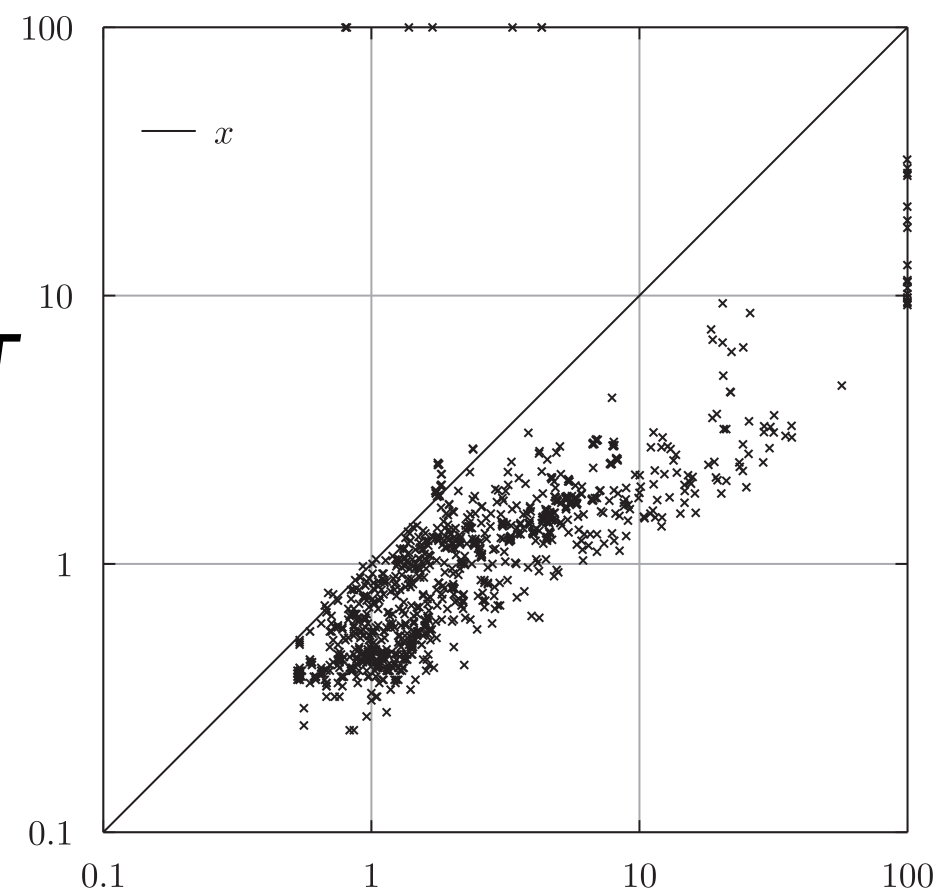
Evaluation

Implemented [Sebastiani & Tomasi, IJCAR'12] within Z3

- *Extended it to optimize multiple objectives simultaneously*

Time in s / benchmark

Our OptMathSAT (in Z3)



Symba (vanilla)

Symba consistently slower than OptMathSAT(Z3)

Evaluation

Optimized Symba

- *Spends 40% of the time (at most) performing unbounded checks*
- *Uses a modified, "locally optimal" Z3 for growing under-approx*

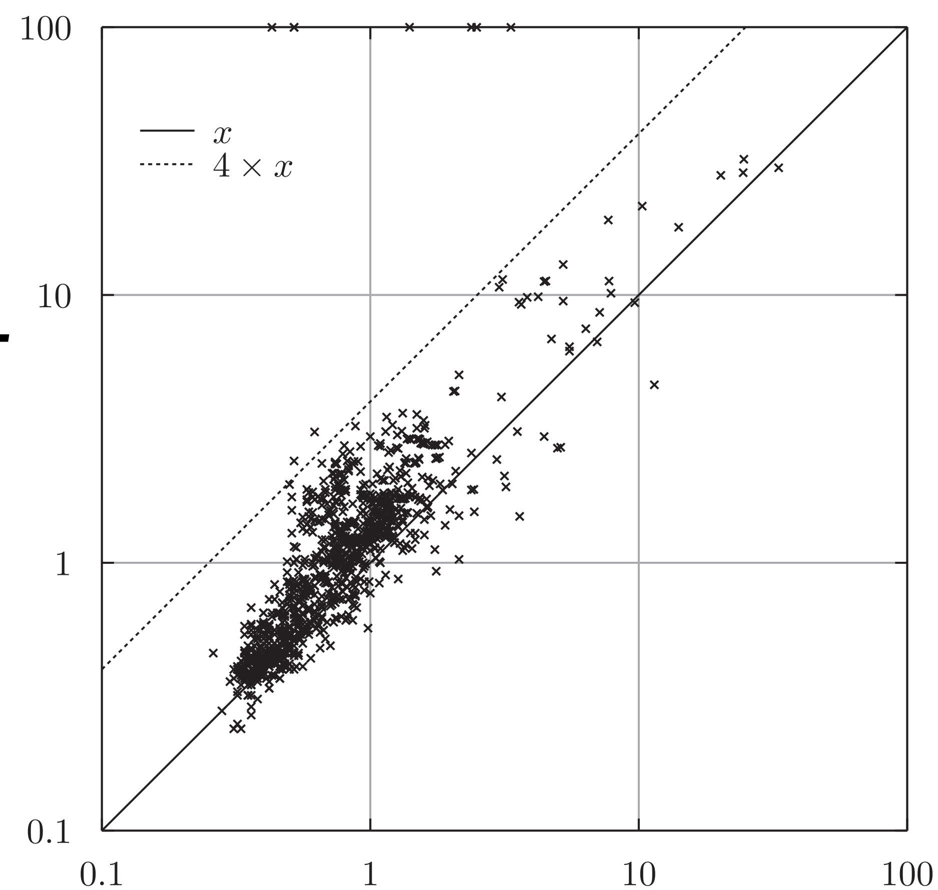
Evaluation

Optimized Symba

- *Spends 40% of the time (at most) performing unbounded checks*
- *Uses a modified, "locally optimal" Z3 for growing under-approx*

Time in s / benchmark

Our OptMathSAT (in Z3)



Symba (optimized)

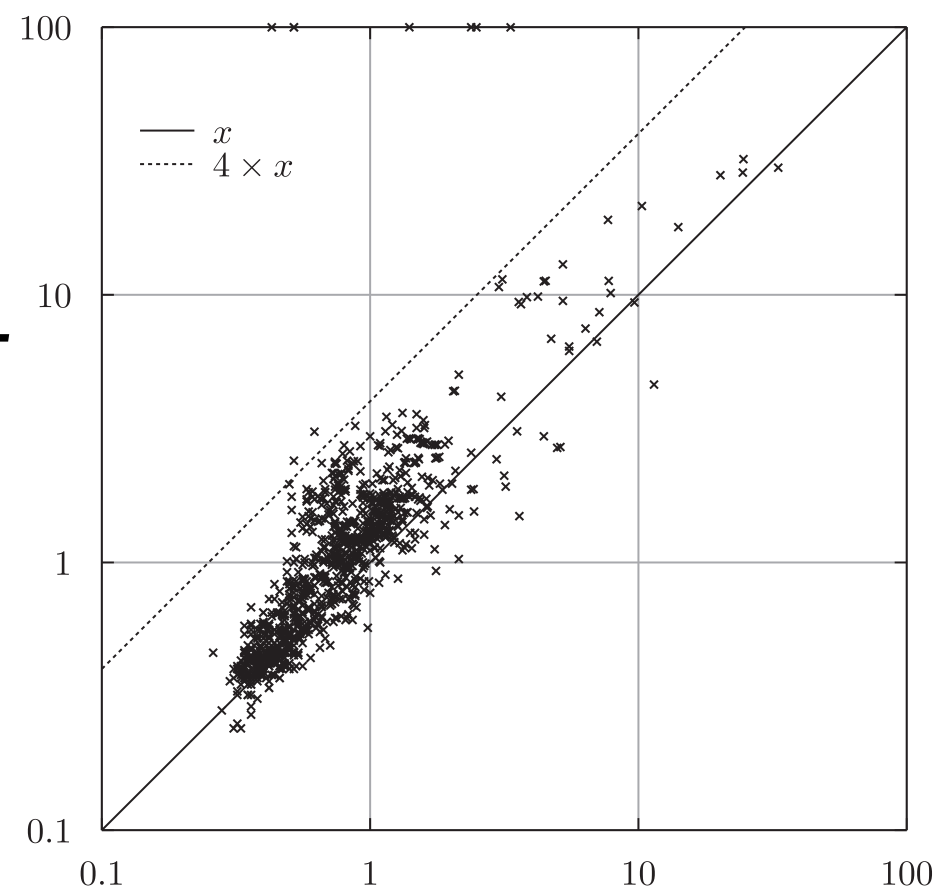
Evaluation

Optimized Symba

- *Spends 40% of the time (at most) performing unbounded checks*
- *Uses a modified, "locally optimal" Z3 for growing under-approx*

Time in s / benchmark

Our OptMathSAT (in Z3)



Symba (optimized)

1.5x speedup over OptMathSAT(Z3)

No timeouts

Best Symba config.
(see paper for more)

Conclusion

Symba: non-convex optimization

- *Efficient SMT-based implementation*
- *Many applications in program analysis and beyond*

Future work

- *Integer arithmetic*
- *Non-linear arithmetic*
- *Parallelization*

Conclusion

`bitbucket.org/arieg/ufo`

Symba: non-convex optimization

- *Efficient SMT-based implementation*
- *Many applications in program analysis and beyond*

Future work

- *Integer arithmetic*
- *Non-linear arithmetic*
- *Parallelization*