Parameter synthesis for algebraic problems with a Boolean structure

Master of Science Thesis

Nicolai Radke
Supervision: Prof. Dr. Erika Ábrahám

RWTH Aachen University
LuFG Theory of Hybrid Systems

WS 2021/2022

Outline

Outline

Outline

1. Preliminaries
2. Parameter Synthesis
   - Base Algorithm
   - Sampling Heuristics
   - Splitting Heuristics
   - Incremental Solving
3. Experimental Evaluation

Outline

Table of Contents

Preliminaries

Nonlinear Real Arithmetic

Preliminaries

Nonlinear Real Arithmetic

- first-order logic over the reals ($\mathbb{R}$)

Preliminaries

Nonlinear Real Arithmetic

- first-order logic over the reals ($\mathbb{R}$)
- functions: $\cdot, +$

Preliminaries

Nonlinear Real Arithmetic

- first-order logic over the reals ($\mathbb{R}$)
- functions: $\cdot, +$
- predicate symbols: $<, \leq, =, \geq, >$

Preliminaries

Nonlinear Real Arithmetic

- first-order logic over the reals ($\mathbb{R}$)
- functions: $\cdot, +$
- predicate symbols: $<, \leq, =, \geq, >$
- quantifiers: $\exists, \forall$

Preliminaries

Nonlinear Real Arithmetic

- first-order logic over the reals ($\mathbb{R}$)
- functions: $\cdot, +$
- predicate symbols: $<, \leq, =, \geq, >$
- quantifiers: $\exists, \forall$

Example

$$\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$$

Preliminaries

SMT-Solving

Preliminaries

SMT-Solving

Input

- logical formula $\varphi(x_1, ..., x_n)$

Preliminaries

SMT-Solving

Input

- logical formula $\varphi(x_1, ..., x_n)$

Output

- Is $\varphi(x_1, ..., x_n)$ satisfiable?

Preliminaries

SMT-Solving

Input

- logical formula $\varphi(x_1, ..., x_n)$

Output

- Is $\varphi(x_1, ..., x_n)$ satisfiable?

Example

Input

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

Preliminaries

SMT-Solving

Input

- logical formula $\varphi(x_1, ..., x_n)$

Output

- Is $\varphi(x_1, ..., x_n)$ satisfiable?

Example

Input

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

Output

- $\varphi(0, 0)$

Preliminaries

SMT-Solving

Input

- logical formula $\varphi(x_1, ..., x_n)$

Output

- Is $\varphi(x_1, ..., x_n)$ satisfiable?

Example

Input

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

Output

- $\varphi(0, 0) = (0 \leq 0) \vee (0 \geq 0^3) = \mathit{true} \vee \mathit{true} = \mathit{true}$

Preliminaries

SMT-Solving

Input

- logical formula $\varphi(x_1, ..., x_n)$

Output

- Is $\varphi(x_1, ..., x_n)$ satisfiable?

Example

Input

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

Output

- $\varphi(0, 0) = (0 \leq 0) \vee (0 \geq 0^3) = \mathit{true} \vee \mathit{true} = \mathit{true}$
- φ is satisfiable!

Preliminaries

Parameter Synthesis Problem

Preliminaries

Parameter Synthesis Problem

Input

- $\varphi(x_1, ..., x_n)$

Preliminaries

Parameter Synthesis Problem

Input

- $\varphi(x_1, ..., x_n)$

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

Preliminaries

Parameter Synthesis Problem

Input

- $\varphi(x_1, ..., x_n)$
- $B_{\text{init}} \subseteq \mathbb{R}^n$

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

Preliminaries

Parameter Synthesis Problem

Input

- $\varphi(x_1, ..., x_n)$
- $B_{\text{init}} \subseteq \mathbb{R}^n$

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$
- $B_{\text{init}} = [-1, 1] \times [-1, 1]$

Preliminaries

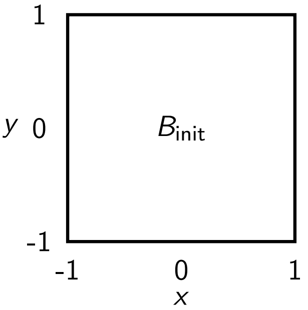
Parameter Synthesis Problem

Input

- $\varphi(x_1, \ldots, x_n)$
- $B_{\text{init}} \subseteq \mathbb{R}^n$

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$
- $B_{\text{init}} = [-1, 1] \times [-1, 1]$



Preliminaries

Parameter Synthesis Problem

Input

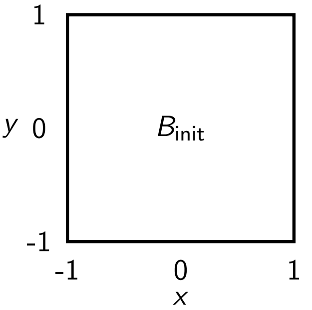
- $\varphi(x_1, \dots, x_n)$
- $B_{\text{init}} \subseteq \mathbb{R}^n$

Goal

- find S_+, S_-
- $S_+ \dot{\cup} S_- = B_{\text{init}}$

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$
- $B_{\text{init}} = [-1, 1] \times [-1, 1]$



Preliminaries

Parameter Synthesis Problem

Input

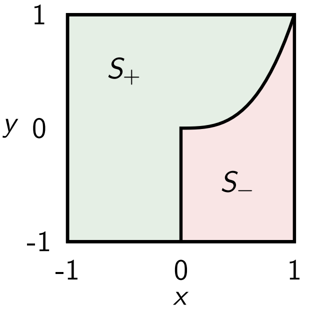
- $\varphi(x_1, ..., x_n)$
- $B_{\text{init}} \subseteq \mathbb{R}^n$

Goal

- find S_+, S_-
- $S_+ \dot{\cup} S_- = B_{\text{init}}$

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$
- $B_{\text{init}} = [-1, 1] \times [-1, 1]$



Preliminaries

Relaxed Parameter Synthesis Problem

Relaxed Parameter Synthesis Problem

Input

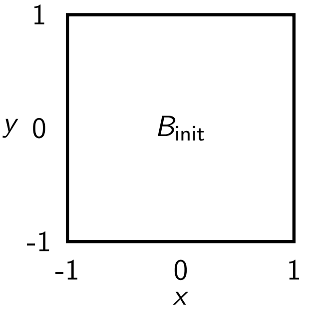
- $\varphi(x_1, ..., x_n)$
- $B_{\text{init}} \subseteq \mathbb{R}^n$

Goal

- find S_+, S_-

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$
- $B_{\text{init}} = [-1, 1] \times [-1, 1]$



Preliminaries

Relaxed Parameter Synthesis Problem

Input

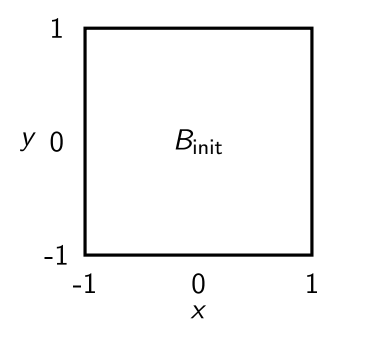
- $\varphi(x_1, \dots, x_n)$
- $B_{\text{init}} \subseteq \mathbb{R}^n$

Goal

- find S_+, S_-
- $\cancel{S_+ \dot{\cup} S_- = B_{\text{init}}}$

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$
- $B_{\text{init}} = [-1, 1] \times [-1, 1]$



Preliminaries

Relaxed Parameter Synthesis Problem

Input

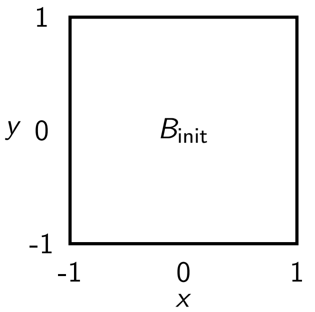
- $\varphi(x_1, \ldots, x_n)$
- $B_{\text{init}} \subseteq \mathbb{R}^n$

Goal

- find S_+, S_-
- $\cancel{S_+ \dot\cup S_- = B_{\text{init}}}$
- $S_+, S_- \subseteq B_{\text{init}}$

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$
- $B_{\text{init}} = [-1, 1] \times [-1, 1]$



Preliminaries

Relaxed Parameter Synthesis Problem

Input

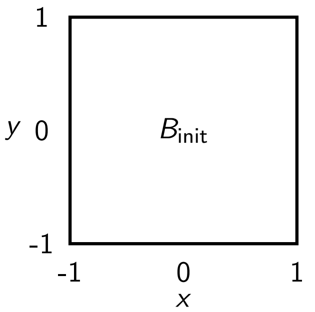
- $\varphi(x_1, \dots, x_n)$
- $B_{\text{init}} \subseteq \mathbb{R}^n$

Goal

- find S_+, S_-
- $\cancel{S_+ \dot\cup S_- = B_{\text{init}}}$
- $S_+, S_- \subseteq B_{\text{init}}$
- $\forall x(x \in S_+ \rightarrow \varphi(x))$

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$
- $B_{\text{init}} = [-1, 1] \times [-1, 1]$



Relaxed Parameter Synthesis Problem

Input

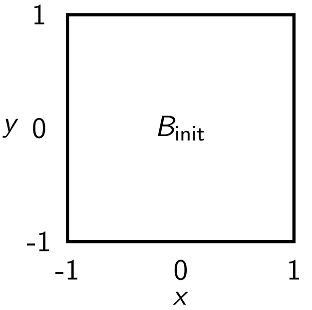
- $\varphi(x_1, \dots, x_n)$
- $B_{\text{init}} \subseteq \mathbb{R}^n$

Goal

- find S_+, S_-
- $\cancel{S_+ \dot{\cup} S_- = B_{\text{init}}}$
- $S_+, S_- \subseteq B_{\text{init}}$
- $\forall x(x \in S_+ \rightarrow \varphi(x))$
- $\forall x(x \in S_- \rightarrow \neg\varphi(x))$

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$
- $B_{\text{init}} = [-1, 1] \times [-1, 1]$



Preliminaries

Relaxed Parameter Synthesis Problem

Input

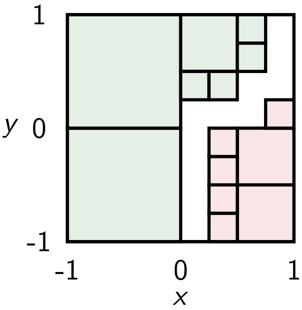
- $\varphi(x_1, \ldots, x_n)$
- $B_{\text{init}} \subseteq \mathbb{R}^n$

Goal

- find S_+, S_-
- $\cancel{S_+ \dot\cup S_- = B_{\text{init}}}$
- $S_+, S_- \subseteq B_{\text{init}}$
- $\forall x(x \in S_+ \rightarrow \varphi(x))$
- $\forall x(x \in S_- \rightarrow \neg\varphi(x))$

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$
- $B_{\text{init}} = [-1, 1] \times [-1, 1]$



Preliminaries

Satisfying Box

Preliminaries

Satisfying Box

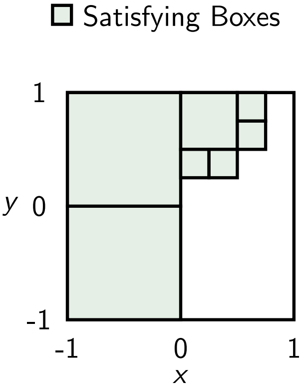
- $\forall x(B(x) \rightarrow \varphi(x))$

Preliminaries

Satisfying Box

- $\forall x(B(x) \rightarrow \varphi(x))$

Example



Preliminaries

Satisfying Box

- $\forall x(B(x) \rightarrow \varphi(x))$

Unsatisfying Box

- $\forall x(B(x) \rightarrow \neg\varphi(x))$

Example

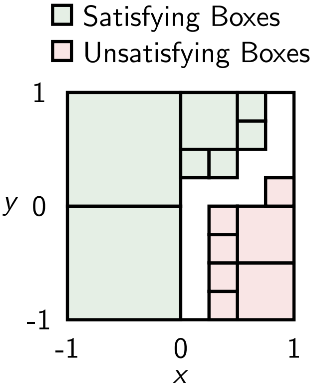


Table of Contents

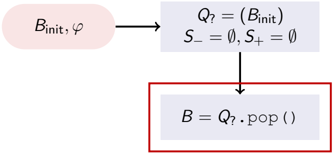
Overview

$\boxed{B_{\text{init}}, \varphi}$

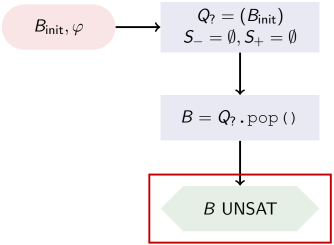
Overview

B_{init}, φ $\longrightarrow$ $\boxed{\begin{array}{c} Q_? = (B_{\text{init}}) \\ S_- = \emptyset, S_+ = \emptyset \end{array}}$

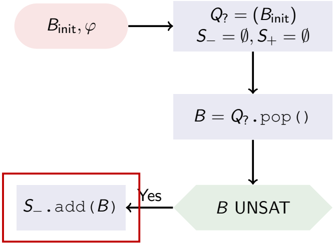
Overview


B_{init}, φ
$Q_? = (B_{\text{init}})$
$S_- = \emptyset, S_+ = \emptyset$
$B = Q_?.\texttt{pop()}$

Overview

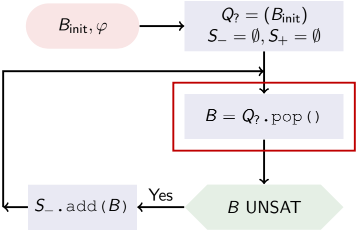

B_{init}, φ
$Q_? = (B_{\text{init}})$
$S_- = \emptyset, S_+ = \emptyset$
$B = Q_?.\texttt{pop()}$
B UNSAT

Overview

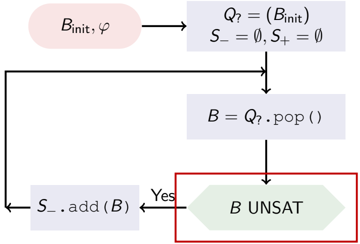


B_{init}, φ
$Q_? = (B_{\text{init}})$
$S_- = \emptyset, S_+ = \emptyset$
$B = Q_?.\texttt{pop()}$
B UNSAT
Yes
$S_-.\texttt{add}(B)$

Overview

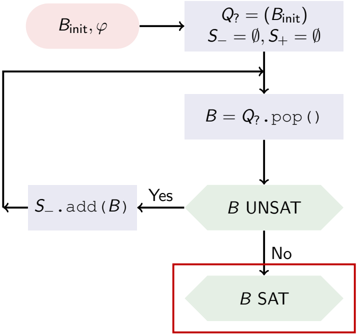

B_{init}, φ
$Q_? = (B_{\text{init}})$
$S_- = \emptyset, S_+ = \emptyset$
$B = Q_?.\texttt{pop()}$
B UNSAT
Yes
$S_-.\texttt{add}(B)$

Overview



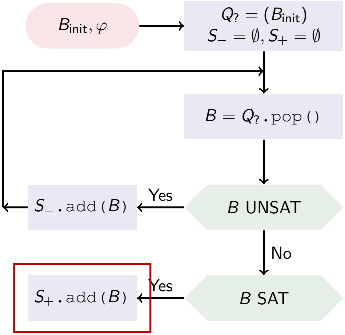
B_{init}, φ
$Q_? = (B_{\text{init}})$
$S_- = \emptyset, S_+ = \emptyset$
$B = Q_?$.pop()
B UNSAT
Yes
S_-.add(B)

Overview



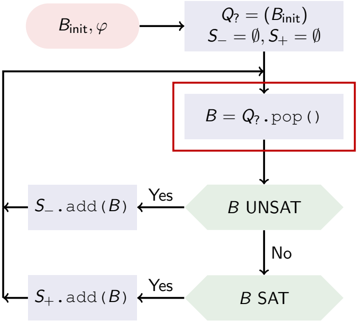
B_{init}, φ
$Q_? = (B_{\text{init}})$
$S_- = \emptyset, S_+ = \emptyset$
$B = Q_?.\texttt{pop()}$
B UNSAT
Yes
$S_-.\texttt{add}(B)$
No
B SAT

Overview



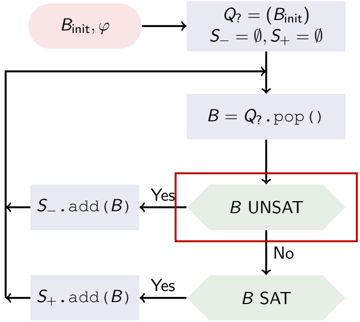
B_{init}, φ
$Q_? = (B_{\text{init}})$
$S_- = \emptyset, S_+ = \emptyset$
$B = Q_?.\texttt{pop()}$
$S_-.\texttt{add}(B)$
Yes
B UNSAT
No
$S_+.\texttt{add}(B)$
Yes
B SAT

Overview



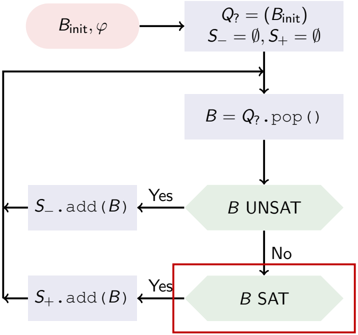
B_{init}, φ
$Q_? = (B_{\text{init}})$
$S_- = \emptyset, S_+ = \emptyset$
$B = Q_?$.pop()
B UNSAT
Yes
S_-.add(B)
No
B SAT
Yes
S_+.add(B)

Overview



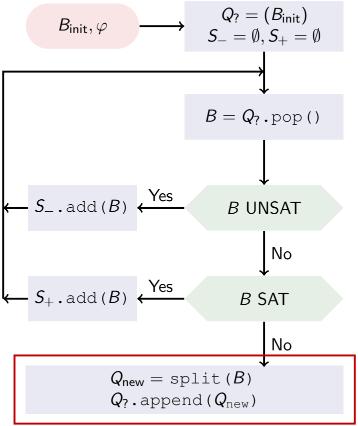
B_{init}, φ
$Q_? = (B_{\text{init}})$
$S_- = \emptyset, S_+ = \emptyset$
$B = Q_?.\texttt{pop()}$
B UNSAT
Yes
$S_-.\texttt{add(}B\texttt{)}$
No
B SAT
Yes
$S_+.\texttt{add(}B\texttt{)}$

Overview



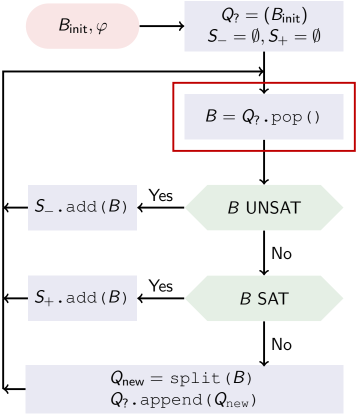
B_{init}, φ
$Q_? = (B_{\text{init}})$
$S_- = \emptyset, S_+ = \emptyset$
$B = Q_?$.pop()
S_-.add(B)
Yes
B UNSAT
No
S_+.add(B)
Yes
B SAT

Overview

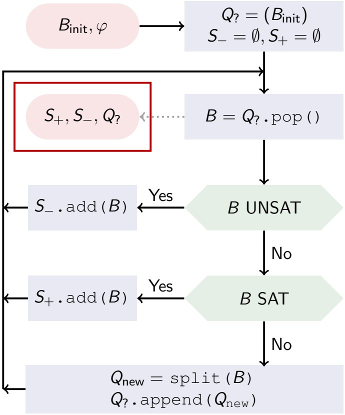


B_{init}, φ
$Q_? = (B_{\text{init}})$
$S_- = \emptyset, S_+ = \emptyset$
$B = Q_?.\texttt{pop()}$
B UNSAT
Yes
$S_-.\texttt{add}(B)$
No
B SAT
Yes
$S_+.\texttt{add}(B)$
No
$Q_{\text{new}} = \texttt{split}(B)$
$Q_?.\texttt{append}(Q_{\text{new}})$

Overview

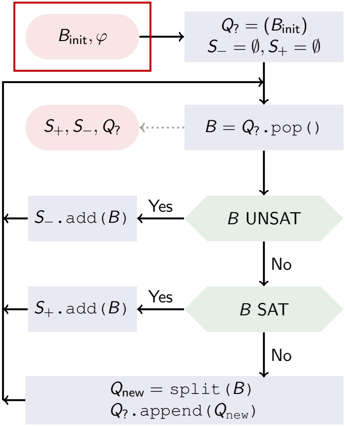

B_{init}, φ
$Q_? = (B_{\text{init}})$
$S_- = \emptyset, S_+ = \emptyset$
$B = Q_?.\texttt{pop()}$
B UNSAT
Yes
$S_-.\texttt{add}(B)$
No
B SAT
Yes
$S_+.\texttt{add}(B)$
No
$Q_{\text{new}} = \texttt{split}(B)$
$Q_?.\texttt{append}(Q_{\texttt{new}})$

Overview


B_{init}, φ
$Q_? = (B_{\text{init}})$
$S_- = \emptyset, S_+ = \emptyset$
$S_+, S_-, Q_?$
$B = Q_?.\texttt{pop()}$
$S_-.\texttt{add}(B)$
Yes
B UNSAT
No
$S_+.\texttt{add}(B)$
Yes
B SAT
No
$Q_{\text{new}} = \texttt{split}(B)$
$Q_?.\texttt{append}(Q_{\text{new}})$

Base Algorithm

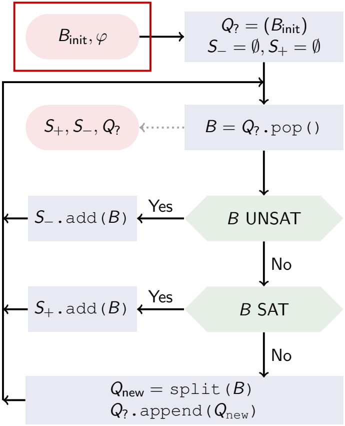
Example



Base Algorithm

Example

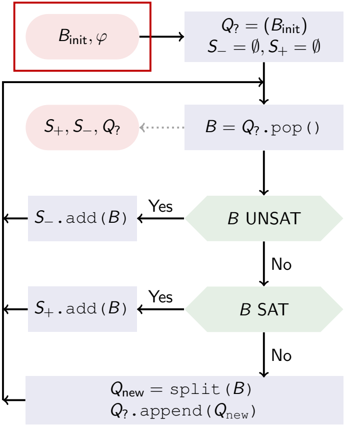
- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$



Base Algorithm

Example

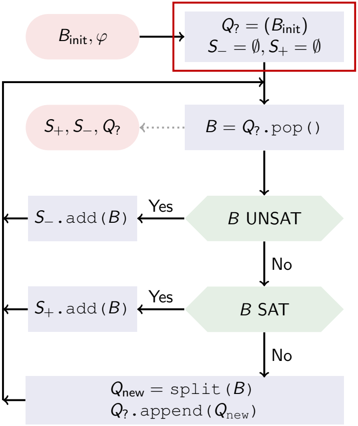
- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$
- $B_{\text{init}} = [-1, 1] \times [-1, 1]$



Base Algorithm

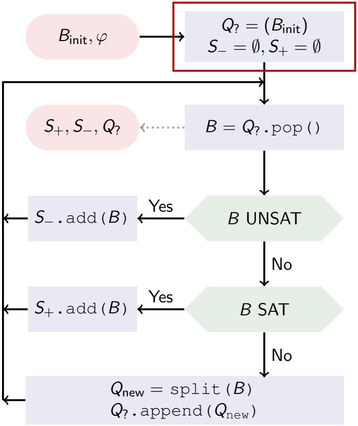
Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$
- $B_{\text{init}} = [-1, 1] \times [-1, 1]$



Example

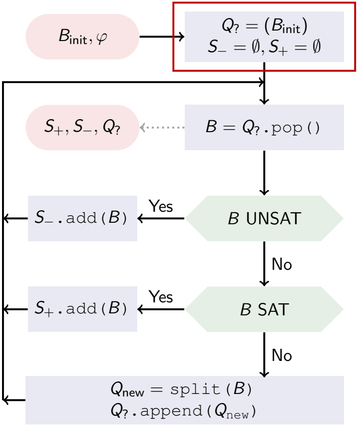
- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$
- $B_{\text{init}} = [-1, 1] \times [-1, 1]$
- $Q_? = (B_{\text{init}})$



Base Algorithm

Example

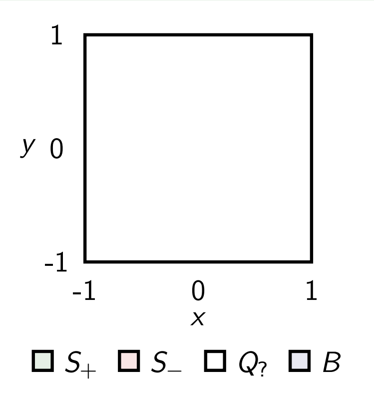
- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$
- $B_{\text{init}} = [-1, 1] \times [-1, 1]$
- $Q_? = (B_{\text{init}})$
- $S_+ = \emptyset$
- $S_- = \emptyset$

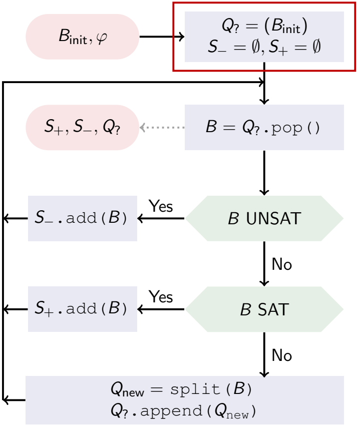


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

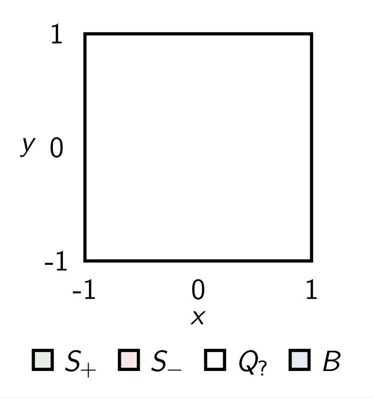


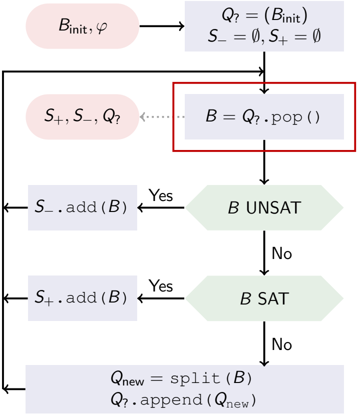


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

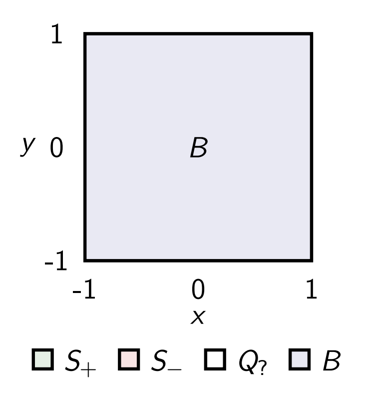


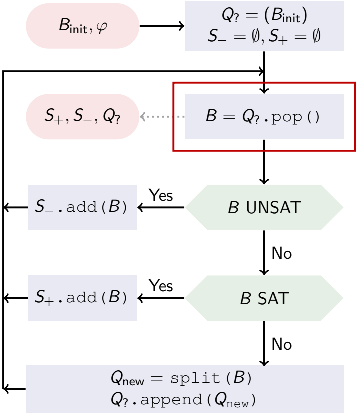


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

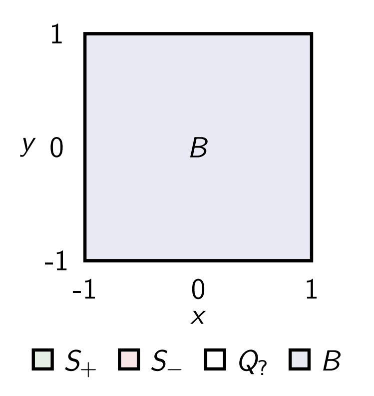


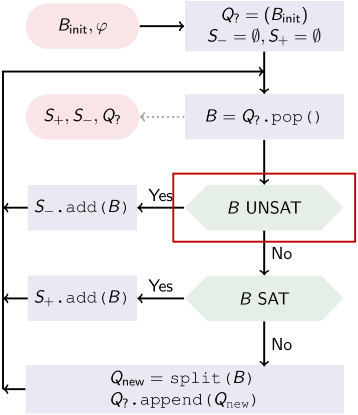


Base Algorithm

Example

- $\varphi(x,y) := (x \leq 0) \vee (y \geq x^3)$

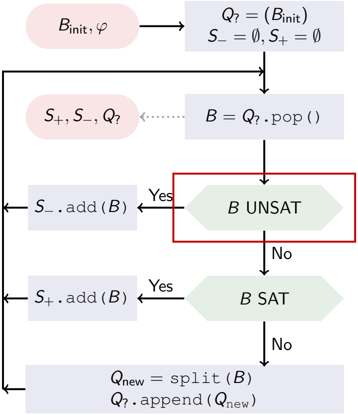




Base Algorithm

Unsatisfying Box

- $\forall x(B(x) \rightarrow \neg\varphi(x))$



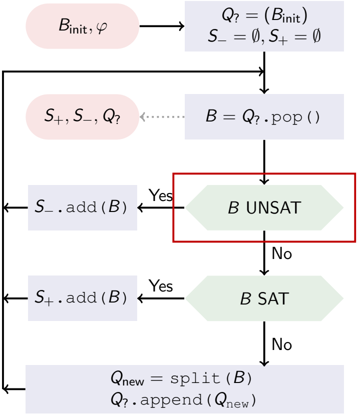
Base Algorithm

Unsatisfying Box

- $\forall x (B(x) \rightarrow \neg\varphi(x))$

Problem

Solvers cannot handle quantifiers.



Base Algorithm

Unsatisfying Box

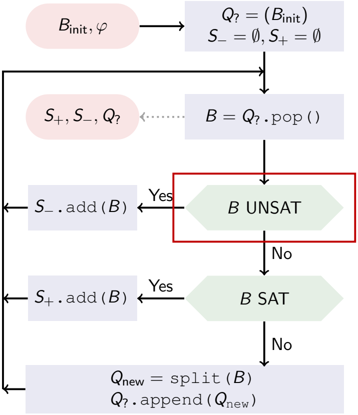
- $\forall x(B(x) \rightarrow \neg\varphi(x))$

Problem

Solvers cannot handle quantifiers.

Solution

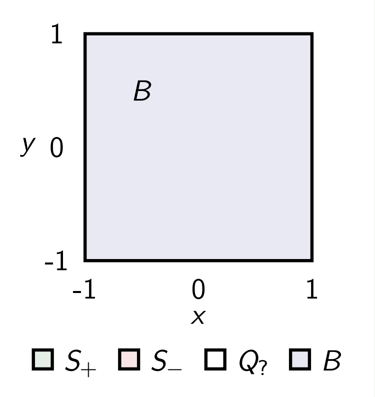
$$\begin{aligned} & \forall x(B(x) \rightarrow \neg\varphi(x)) \\ \equiv\ & \neg\exists x\ \neg(\neg B(x) \vee \neg\varphi(x)) \\ \equiv\ & \neg\exists x\ (B(x) \wedge \varphi(x)) \\ \equiv\ & B(x) \wedge \varphi(x) \text{ is UNSAT} \\ \equiv\ & \text{'No satisfying } x \text{ exists in B'} \end{aligned}$$

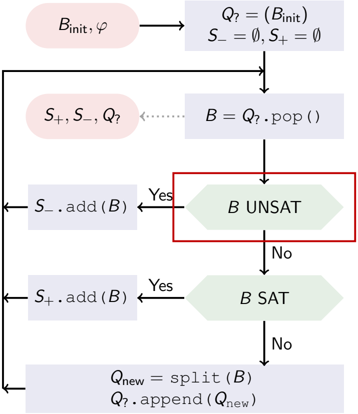


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

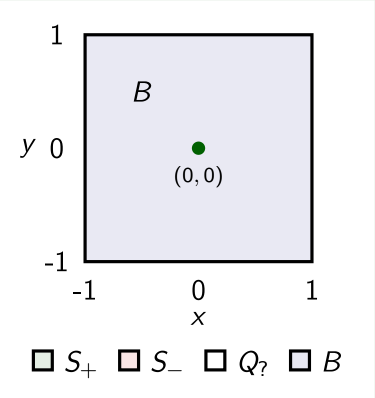


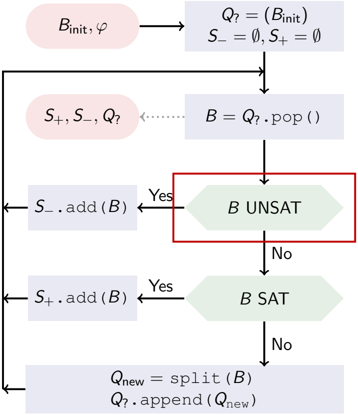


Base Algorithm

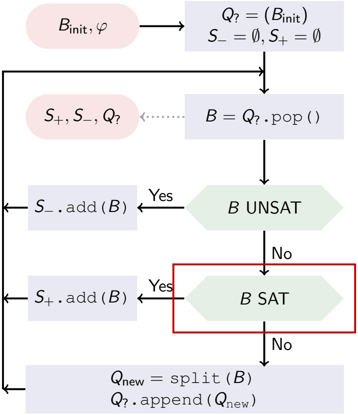
Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$





Base Algorithm



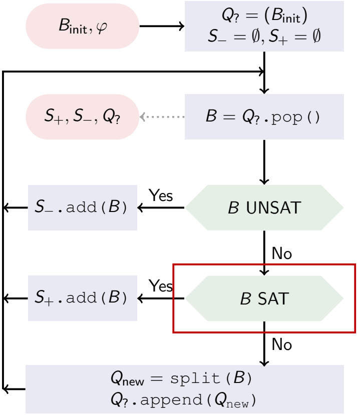
Base Algorithm

Satisfying Box

- $\forall x(B(x) \to \varphi(x))$

Problem

Solvers cannot handle quantifiers.



Base Algorithm

Satisfying Box

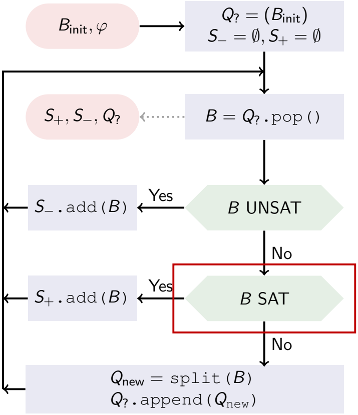
- $\forall x(B(x) \to \varphi(x))$

Problem

Solvers cannot handle quantifiers.

Solution

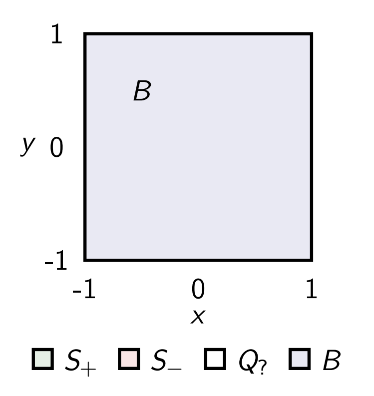
$$\begin{aligned} & \forall x(B(x) \to \varphi(x)) \\ \equiv\ & \neg\exists x\ \neg(\neg B(x) \vee \varphi(x)) \\ \equiv\ & \neg\exists x\ (B(x) \wedge \neg\varphi(x)) \\ \equiv\ & B(x) \wedge \neg\varphi(x) \text{ is UNSAT} \\ \equiv\ & \text{'No unsatisfying } x \text{ exists in B'} \end{aligned}$$

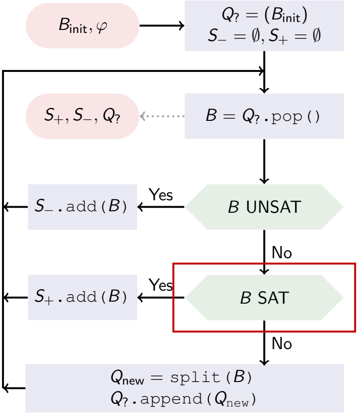


Base Algorithm

Example

- $\varphi(x,y) := (x \leq 0) \vee (y \geq x^3)$

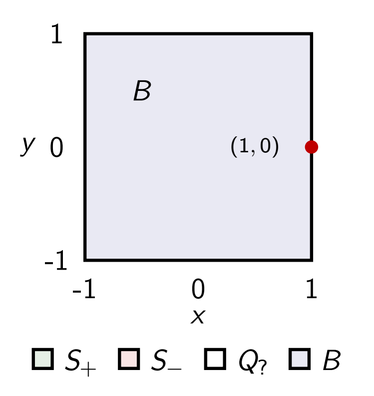


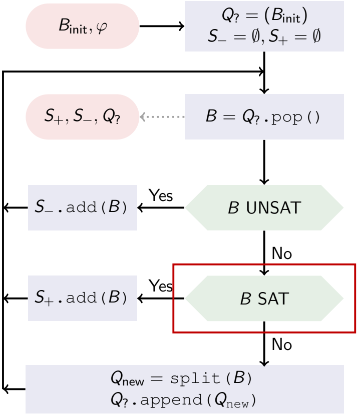


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

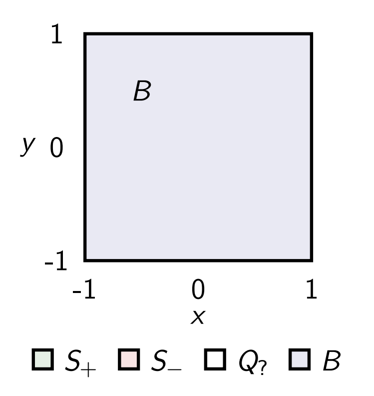


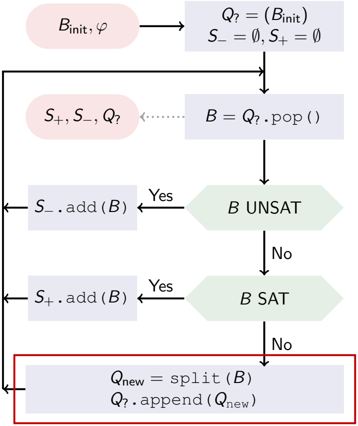


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

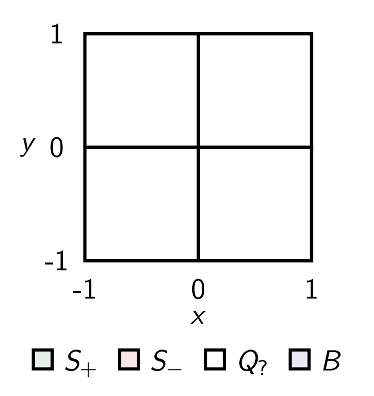


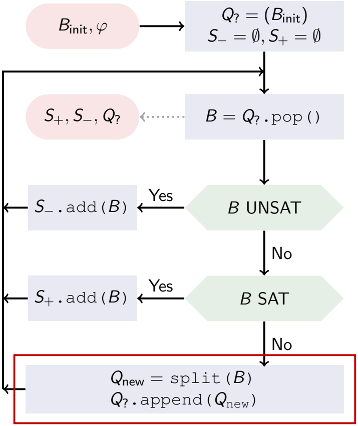


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

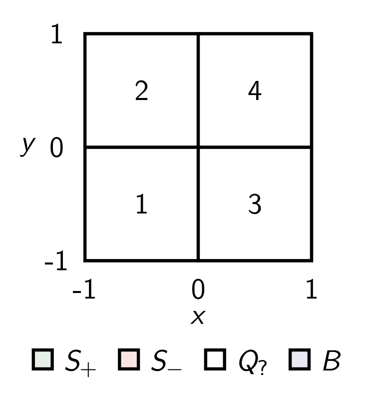


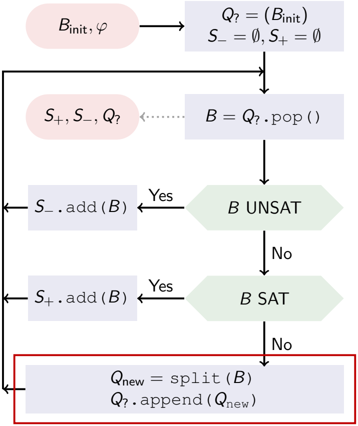


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

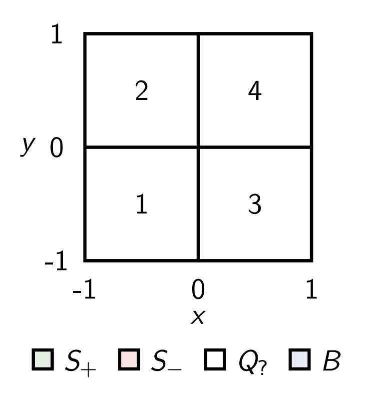


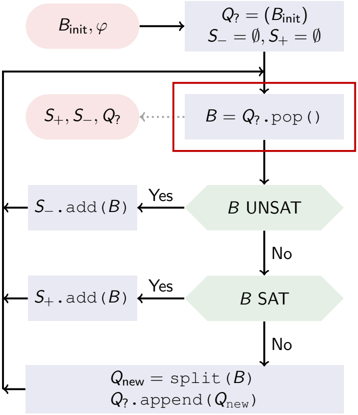


Base Algorithm

Example

- $\varphi(x,y) := (x \leq 0) \vee (y \geq x^3)$

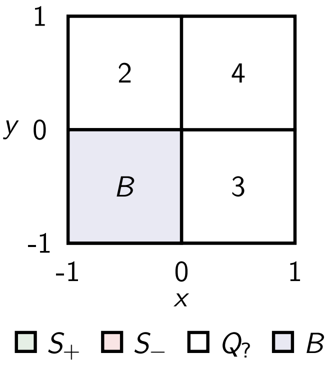


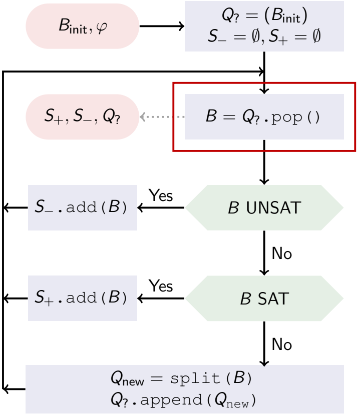


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

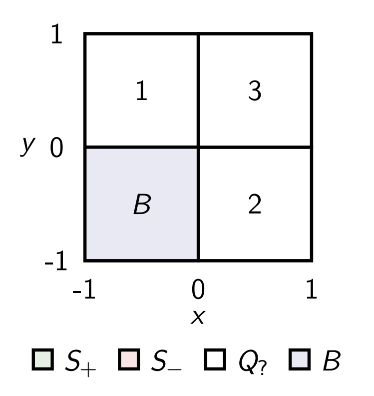


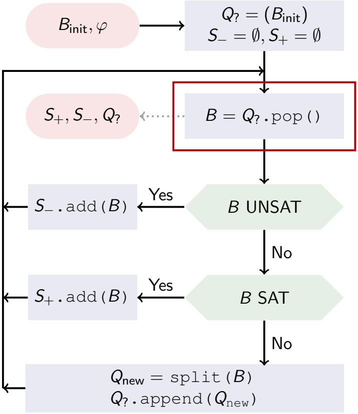


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

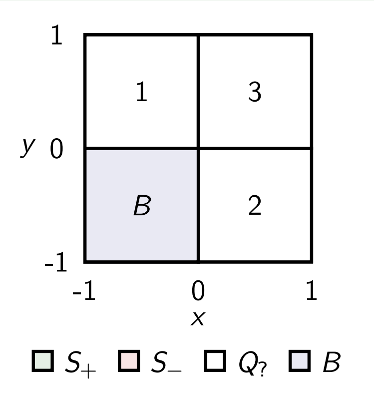


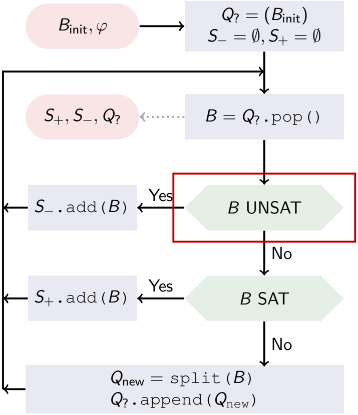


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

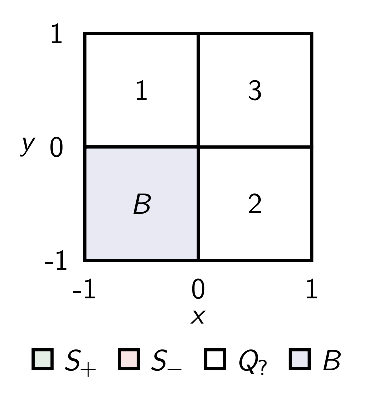


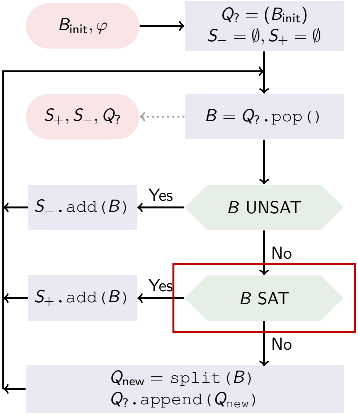


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

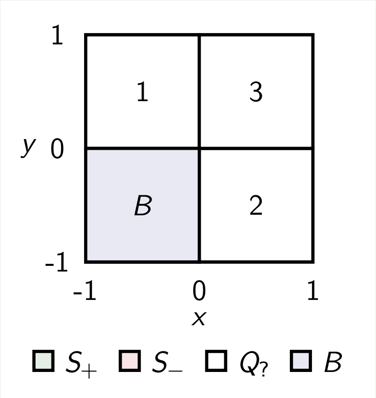


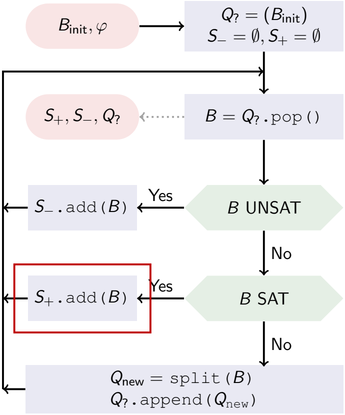


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

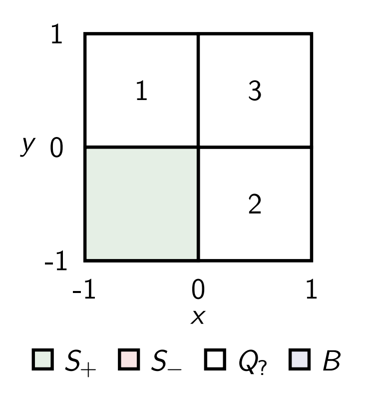


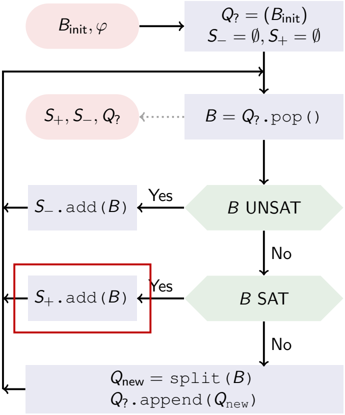


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

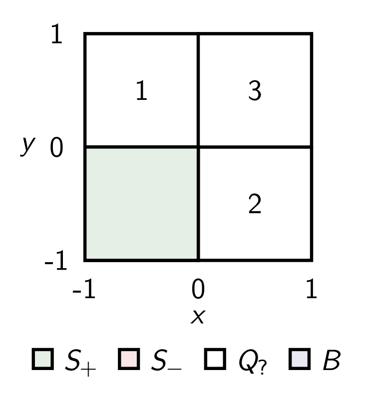


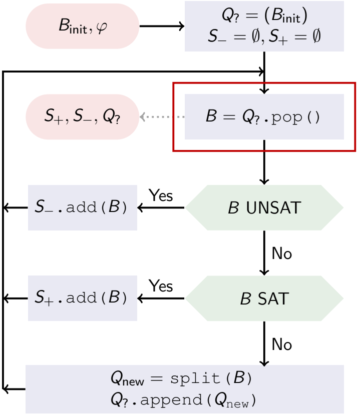


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

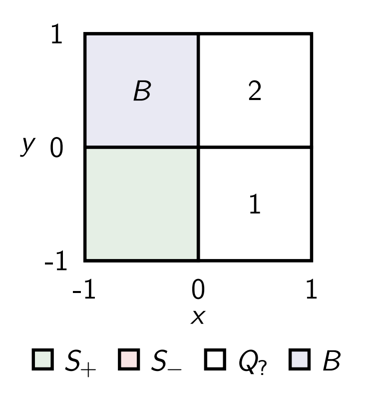


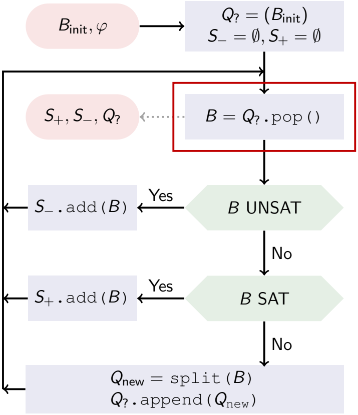


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

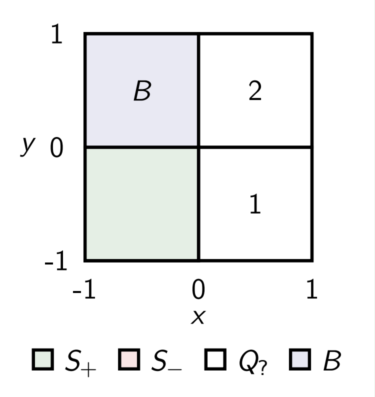


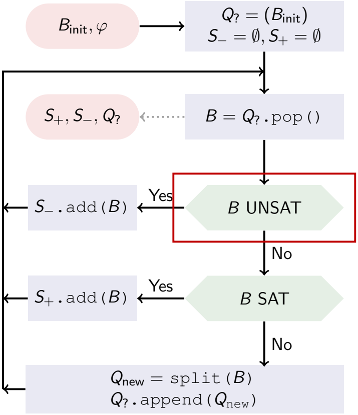


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

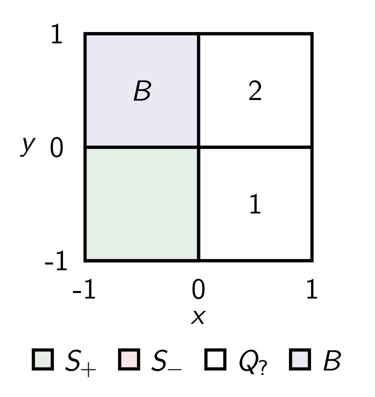


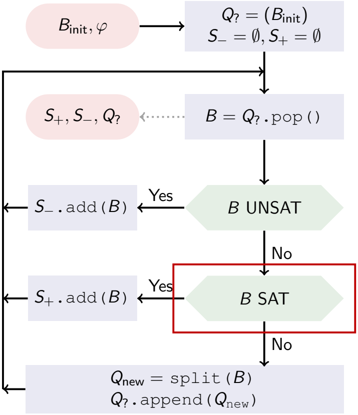


Base Algorithm

Example

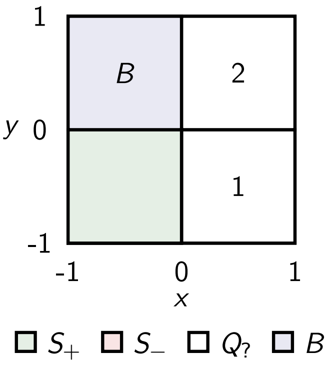
- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

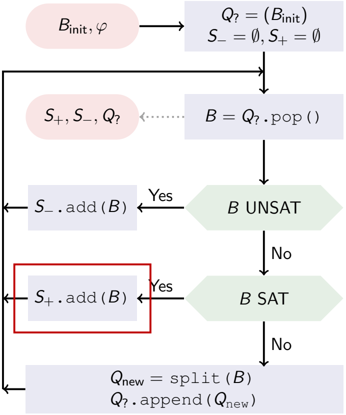




Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

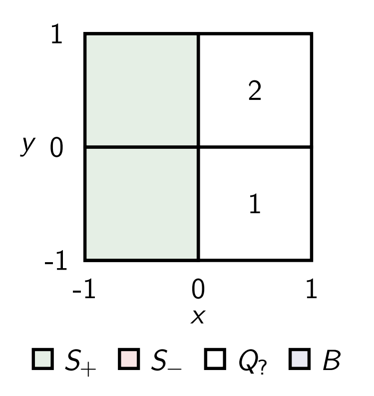


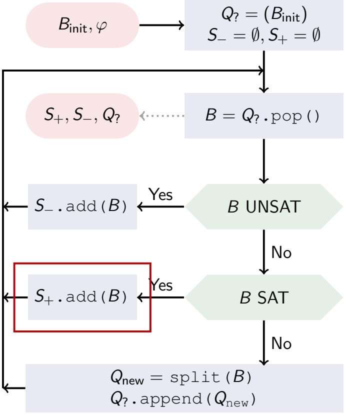


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

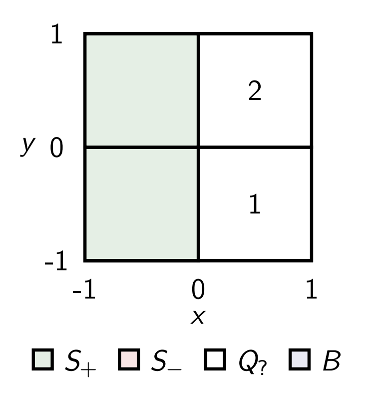


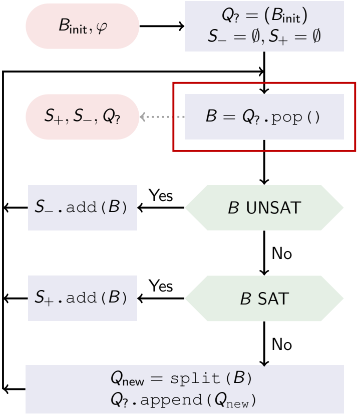


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

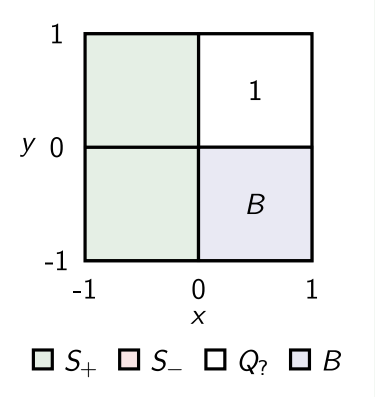


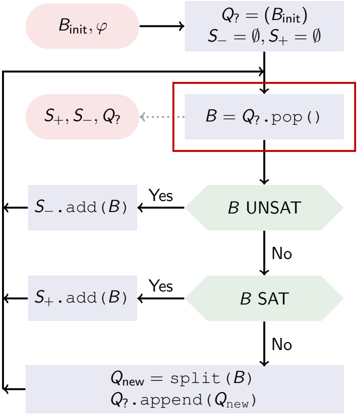


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

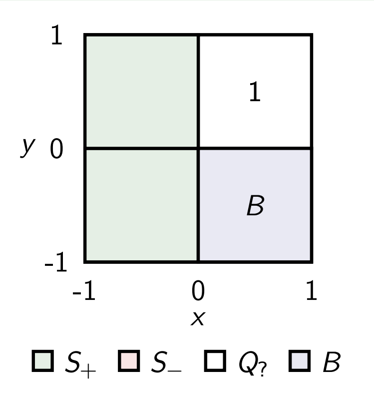


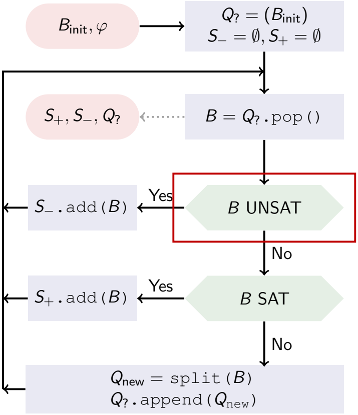


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

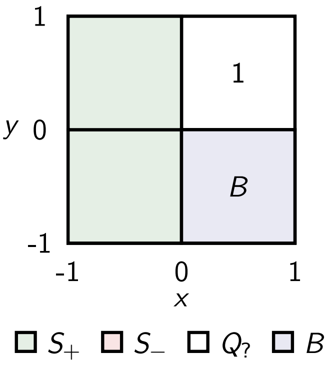


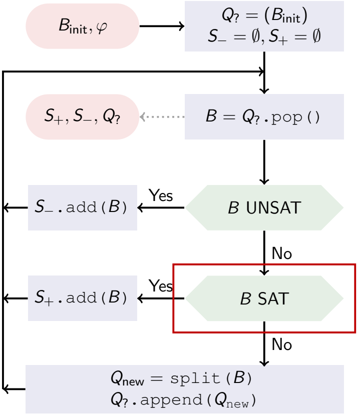


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

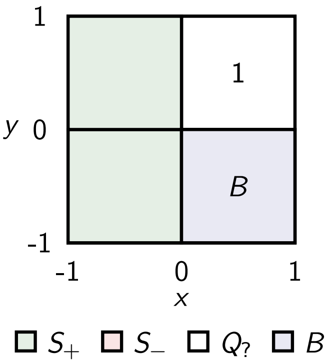


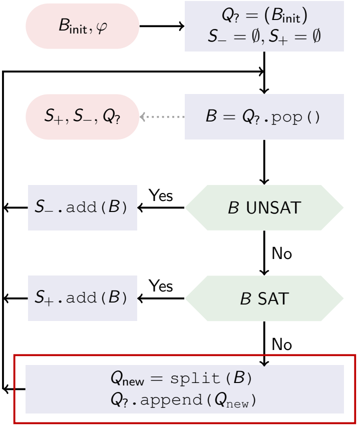


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

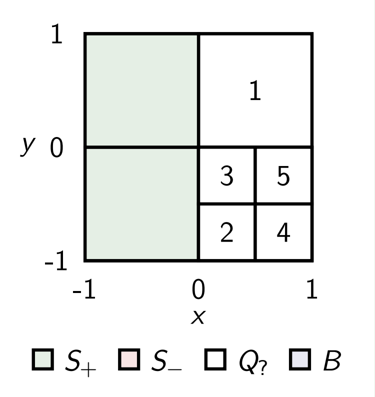


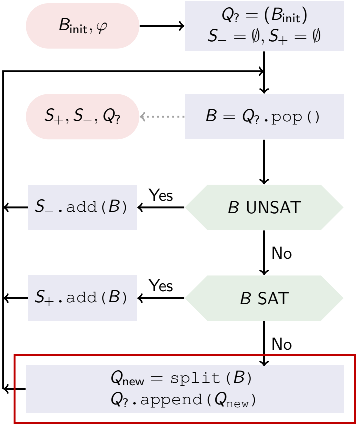


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

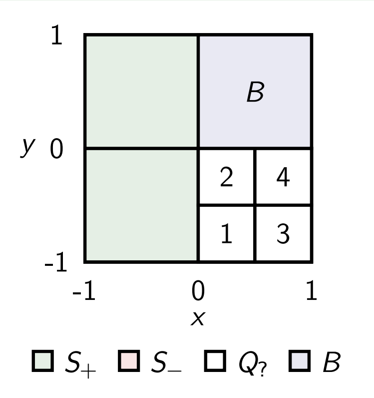


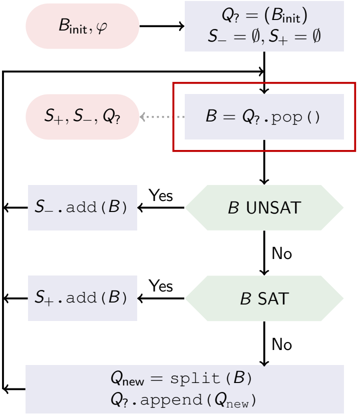


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

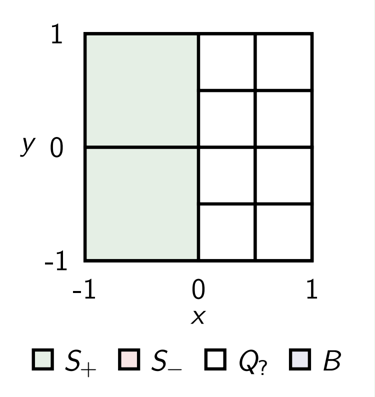


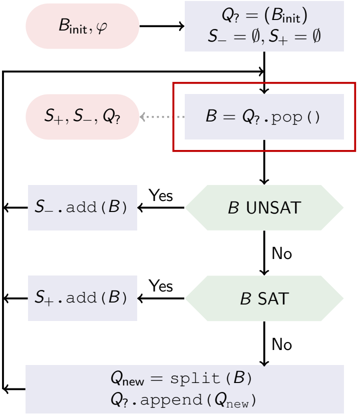


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

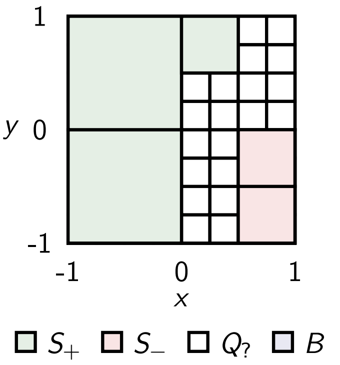


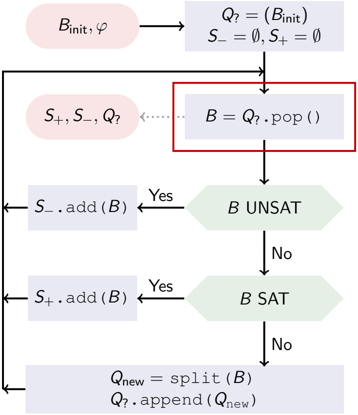


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \lor (y \geq x^3)$

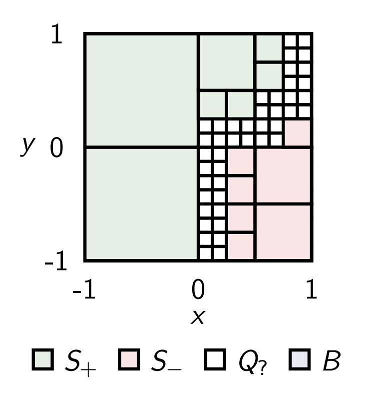


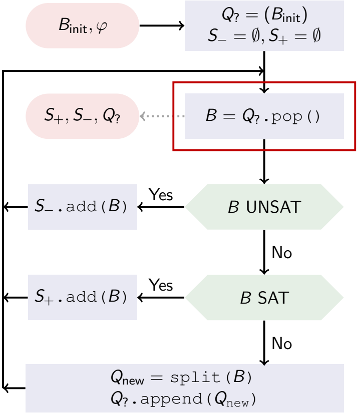


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

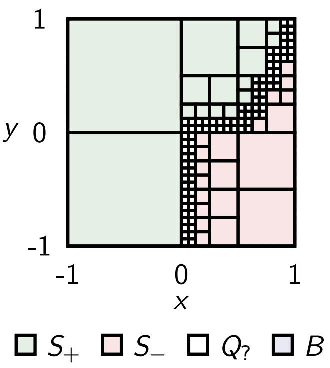


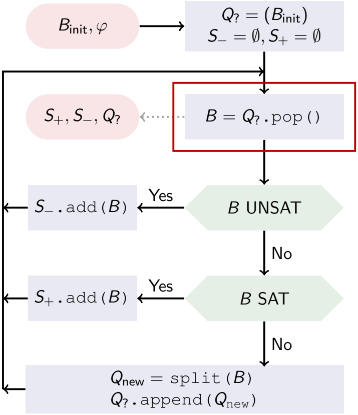


Base Algorithm

Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$

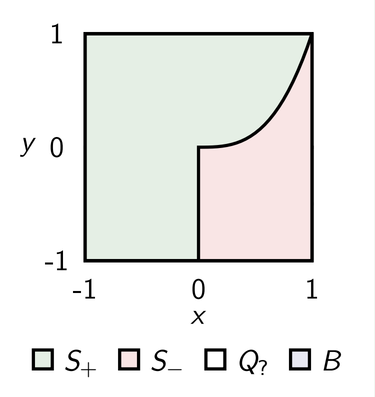


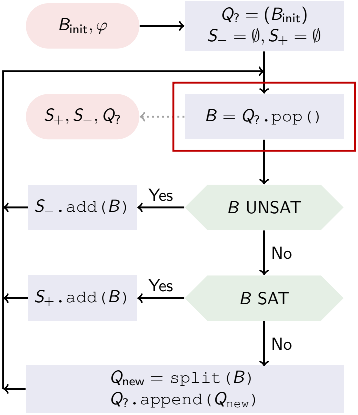


Base Algorithm

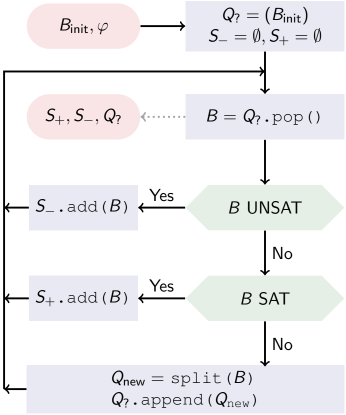
Example

- $\varphi(x, y) := (x \leq 0) \vee (y \geq x^3)$



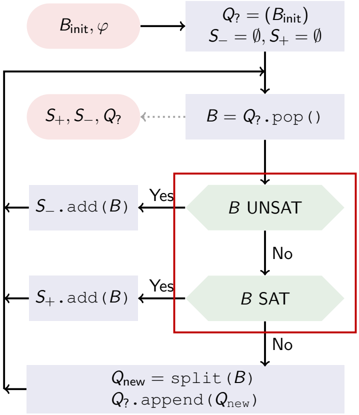


Sampling



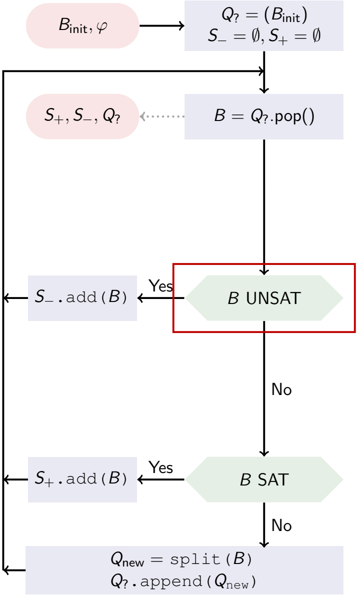
B_{init}, φ
$Q_? = (B_{\text{init}})$
$S_- = \emptyset, S_+ = \emptyset$
$S_+, S_-, Q_?$
$B = Q_?.\texttt{pop()}$
$S_-.\texttt{add}(B)$
Yes
B UNSAT
No
$S_+.\texttt{add}(B)$
Yes
B SAT
No
$Q_{\text{new}} = \texttt{split}(B)$
$Q_?.\texttt{append}(Q_{\text{new}})$

Sampling



$>$ 80% of the running time

Sampling: Recall



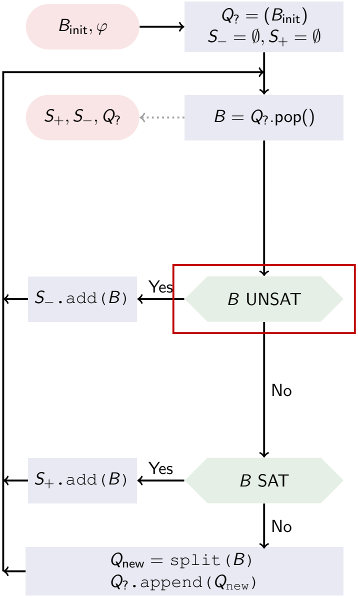
Sampling: Recall

Unsatisfying Box

- $\forall x(B(x) \rightarrow \neg\varphi(x))$

Problem

Solvers cannot handle quantifiers.



Sampling: Recall

Unsatisfying Box

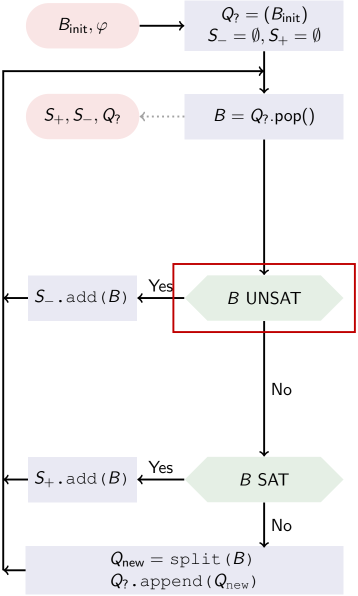
- $\forall x(B(x) \to \neg\varphi(x))$

Problem

Solvers cannot handle quantifiers.

Solution

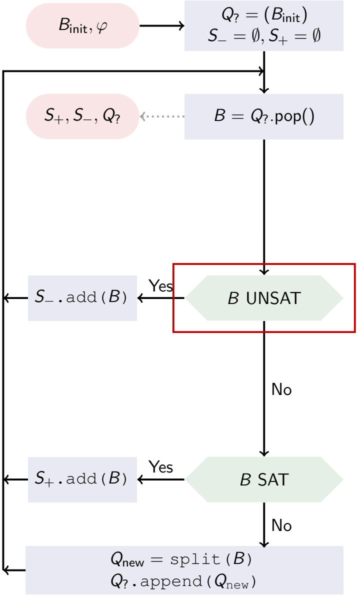
$$
\begin{aligned}
& \forall x(B(x) \to \neg\varphi(x)) \\
\equiv\ & \neg\exists x\ \neg(\neg B(x) \vee \neg\varphi(x)) \\
\equiv\ & \neg\exists x\ (B(x) \wedge \varphi(x)) \\
\equiv\ & B(x) \wedge \varphi(x) \text{ is UNSAT} \\
\equiv\ & \text{'no satisfying } x \text{ exists in B'}
\end{aligned}
$$



Sampling: Idea

Solution

$$\forall x(B(x) \to \neg\varphi(x))$$
$$\equiv B(x) \land \varphi(x) \text{ is UNSAT}$$
$$\equiv \text{'no satisfying } x \text{ exists in B'}$$



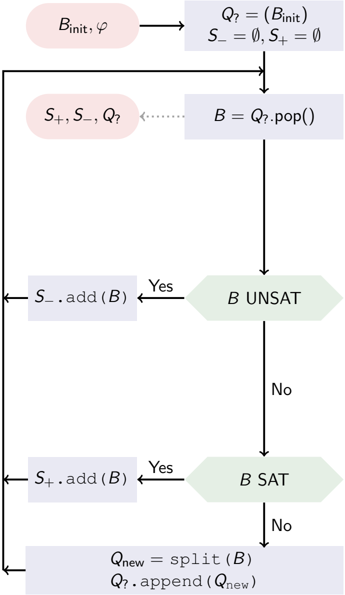
Sampling: Idea

Solution

$$\forall x(B(x) \to \neg\varphi(x))$$
$$\equiv B(x) \land \varphi(x) \text{ is UNSAT}$$
$$\equiv \text{'no satisfying } x \text{ exists in B'}$$

Sampling

- take $x \in B$, check $\varphi(x)$



Sampling: Idea

Solution

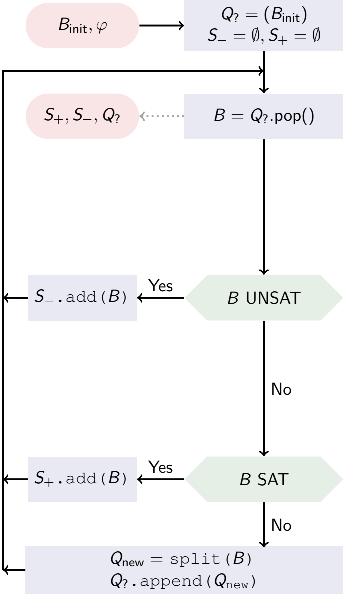
$$\forall x(B(x) \rightarrow \neg\varphi(x))$$

$$\equiv B(x) \wedge \varphi(x) \text{ is UNSAT}$$

$\equiv$ 'no satisfying x exists in B'

Sampling

- take $x \in B$, check $\varphi(x)$
- $\varphi(x)$ holds: B can not be unsatisfying



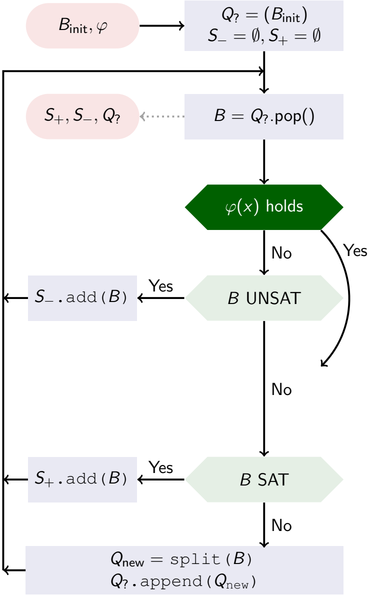
Sampling: Idea

Solution

$$\forall x(B(x) \to \neg\varphi(x))$$
$$\equiv B(x) \land \varphi(x) \text{ is UNSAT}$$
$$\equiv \text{'no satisfying } x \text{ exists in B'}$$

Sampling

- take $x \in B$, check $\varphi(x)$
- $\varphi(x)$ holds: B can not be unsatisfying



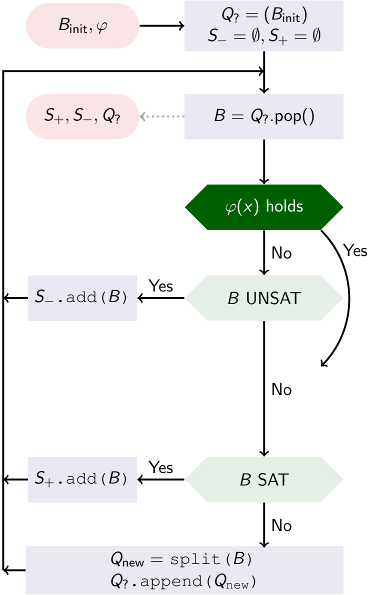
Sampling: Idea

Solution

$$\forall x(B(x) \rightarrow \neg\varphi(x))$$
$$\equiv B(x) \wedge \varphi(x) \text{ is UNSAT}$$
$$\equiv \text{'no satisfying } x \text{ exists in B'}$$

Sampling

- take $x \in B$, check $\varphi(x)$
- $\varphi(x)$ holds: B can not be unsatisfying
- $\varphi(x)$ does not hold: B can not be satisfying



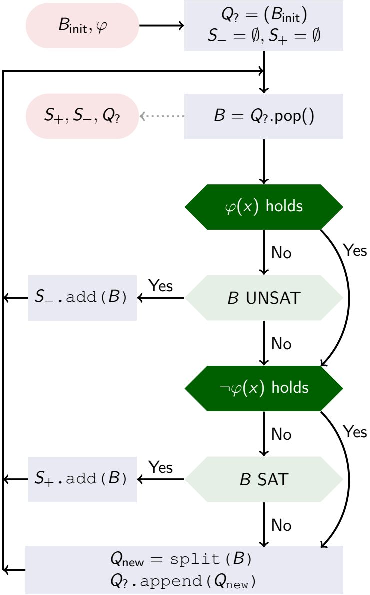
Sampling: Idea

Solution

$$\forall x(B(x) \rightarrow \neg\varphi(x))$$
$$\equiv B(x) \wedge \varphi(x) \text{ is UNSAT}$$
$$\equiv \text{'no satisfying } x \text{ exists in B'}$$

Sampling

- take $x \in B$, check $\varphi(x)$
- $\varphi(x)$ holds: B can not be unsatisfying
- $\varphi(x)$ does not hold: B can not be satisfying



Sampling: Options

Sampling: Options

Normal

Sampling: Options

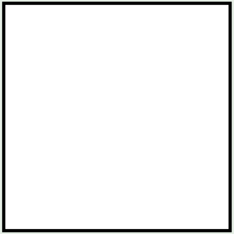
Normal

- sample 'arbitrary' point

Sampling: Options

Normal

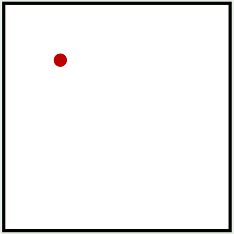
- sample 'arbitrary' point



Sampling: Options

Normal

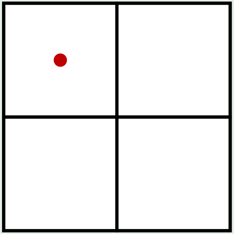
- sample 'arbitrary' point



Sampling: Options

Normal

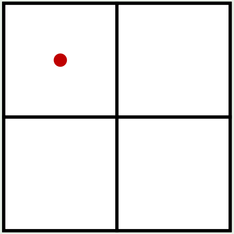
- sample 'arbitrary' point



Sampling: Options

Normal

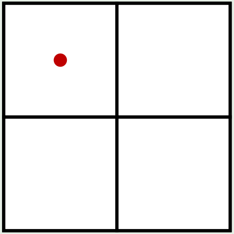
- sample 'arbitrary' point
- optionally carry samples when splitting the box



Sampling: Options

Normal

- sample 'arbitrary' point
- optionally carry samples when splitting the box

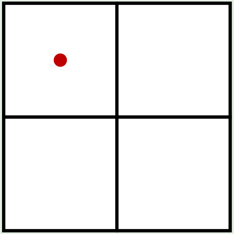


– carrying is time consuming

Sampling: Options

Normal

- sample 'arbitrary' point
- optionally carry samples when splitting the box



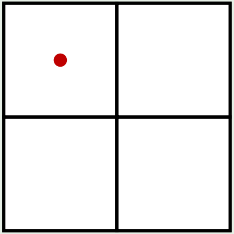
– carrying is time consuming

Clever

Sampling: Options

Normal

- sample 'arbitrary' point
- optionally carry samples when splitting the box



– carrying is time consuming

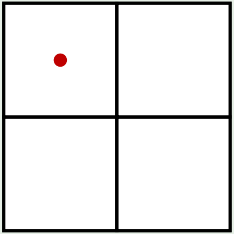
Clever

- take sample on cut

Sampling: Options

Normal

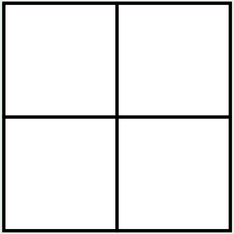
- sample 'arbitrary' point
- optionally carry samples when splitting the box



– carrying is time consuming

Clever

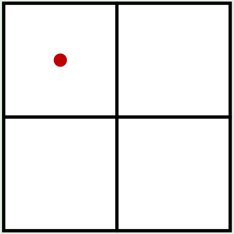
- take sample on cut



Sampling: Options

Normal

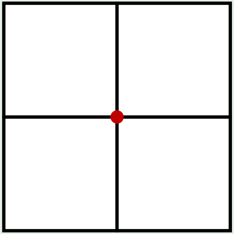
- sample 'arbitrary' point
- optionally carry samples when splitting the box



– carrying is time consuming

Clever

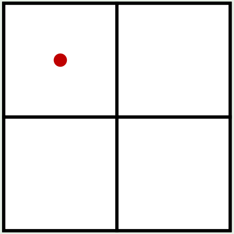
- take sample on cut



Sampling: Options

Normal

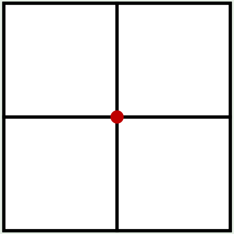
- sample 'arbitrary' point
- optionally carry samples when splitting the box



– carrying is time consuming

Clever

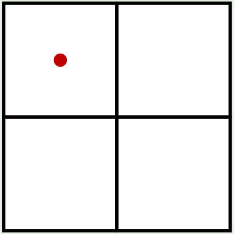
- take sample on cut
- automatically assign the sample to adjacent boxes



Sampling: Options

Normal

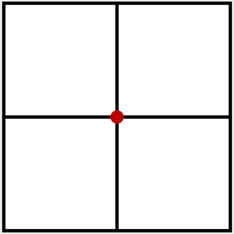
- sample 'arbitrary' point
- optionally carry samples when splitting the box



– carrying is time consuming

Clever

- take sample on cut
- automatically assign the sample to adjacent boxes

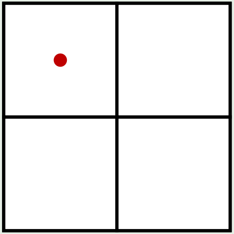


\+ sample only every 2nd split

Sampling: Options

Normal

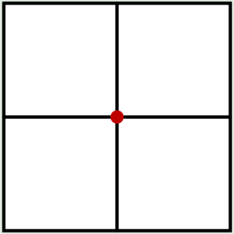
- sample 'arbitrary' point
- optionally carry samples when splitting the box



– carrying is time consuming

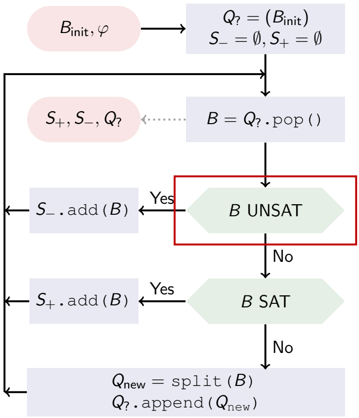
Clever

- take sample on cut
- automatically assign the sample to adjacent boxes



+ sample only every 2nd split
– only works on cuts

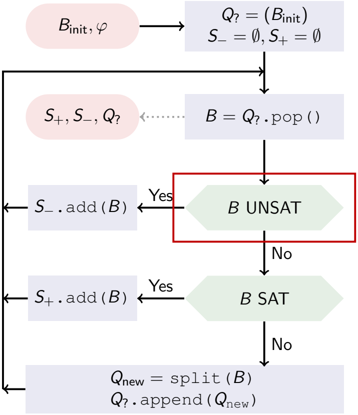
Model Saving



Model Saving

Idea

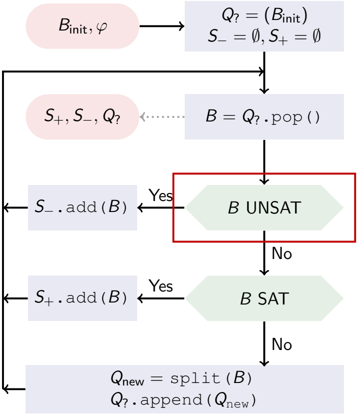
- if the answer is 'No', the solver has found a model



Model Saving

Idea

- if the answer is 'No', the solver has found a model
- use it this model as sample

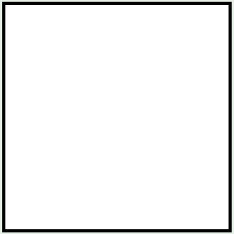


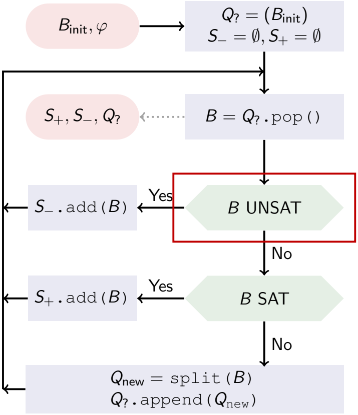
Model Saving

Idea

- if the answer is 'No', the solver has found a model
- use it this model as sample

Example



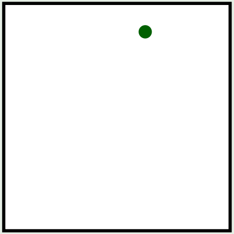


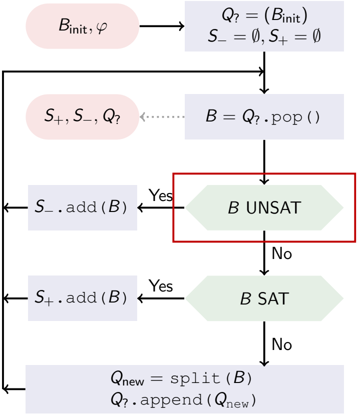
Model Saving

Idea

- if the answer is 'No', the solver has found a model
- use it this model as sample

Example



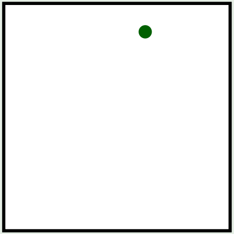


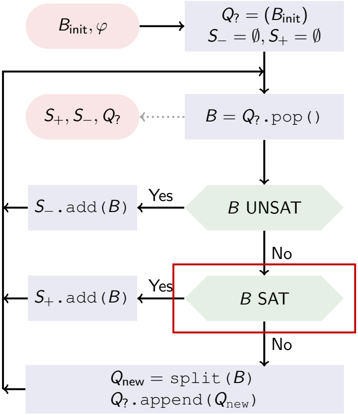
Model Saving

Idea

- if the answer is 'No', the solver has found a model
- use it this model as sample

Example



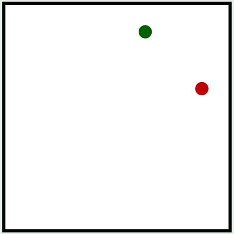


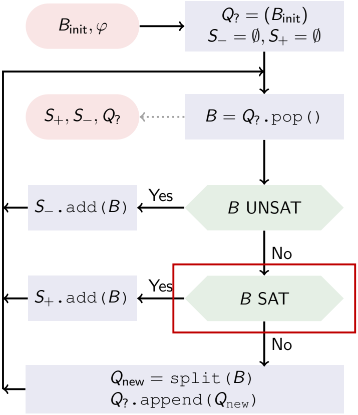
Model Saving

Idea

- if the answer is 'No', the solver has found a model
- use it this model as sample

Example



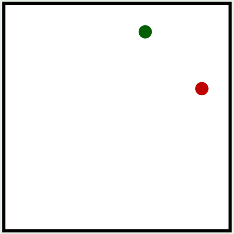


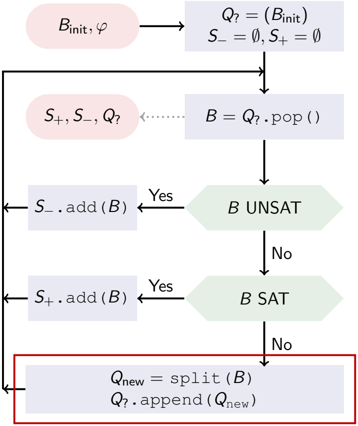
Model Saving

Idea

- if the answer is 'No', the solver has found a model
- use it this model as sample

Example



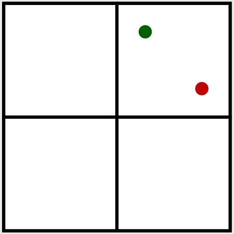


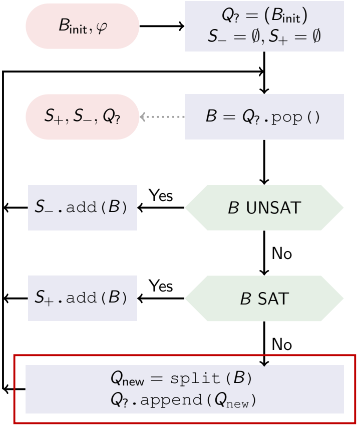
Model Saving

Idea

- if the answer is 'No', the solver has found a model
- use it this model as sample

Example



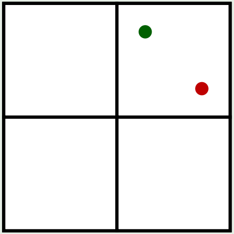


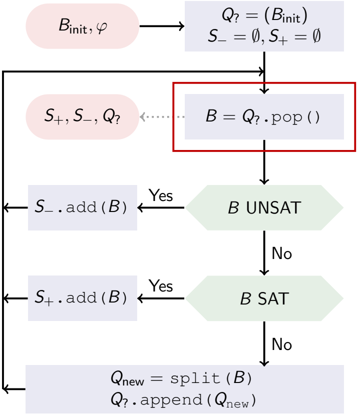
Model Saving

Idea

- if the answer is 'No', the solver has found a model
- use it this model as sample

Example



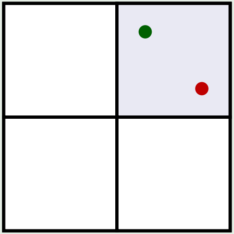


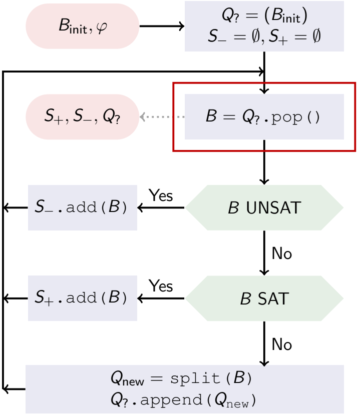
Model Saving

Idea

- if the answer is 'No', the solver has found a model
- use it this model as sample

Example



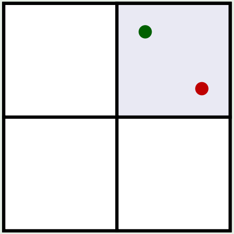


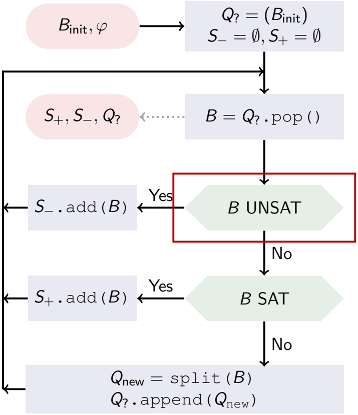
Model Saving

Idea

- if the answer is 'No', the solver has found a model
- use it this model as sample

Example



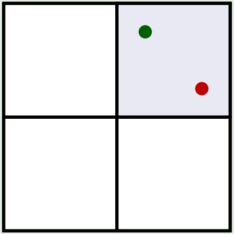


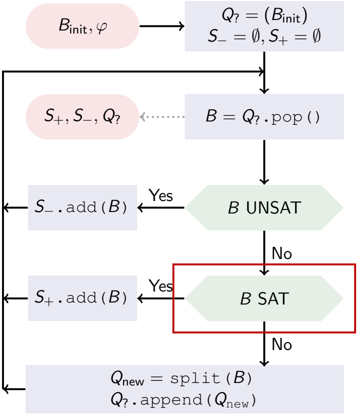
Model Saving

Idea

- if the answer is 'No', the solver has found a model
- use it this model as sample

Example



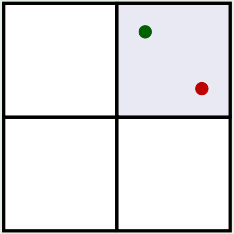


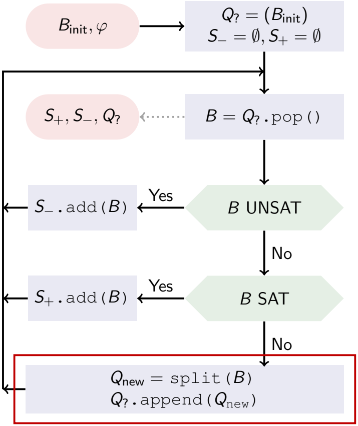
Model Saving

Idea

- if the answer is 'No', the solver has found a model
- use it this model as sample

Example



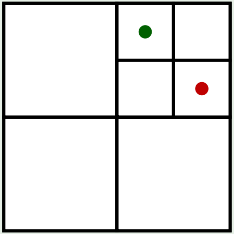


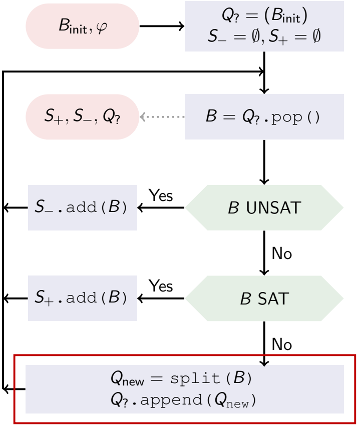
Model Saving

Idea

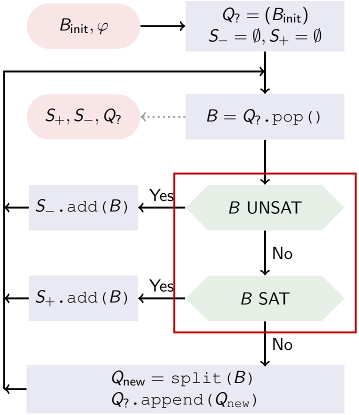
- if the answer is 'No', the solver has found a model
- use it this model as sample

Example





Splitting



$>$ 80% of the running time

Splitting: Options

Bisect All Dimensions

Splitting: Options

Bisect All Dimensions

- d cuts for d dimensions

Splitting: Options

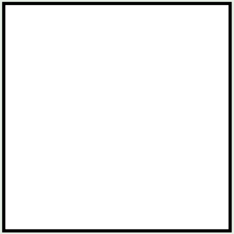
Bisect All Dimensions

- d cuts for d dimensions
- 2^d new boxes

Splitting: Options

Bisect All Dimensions

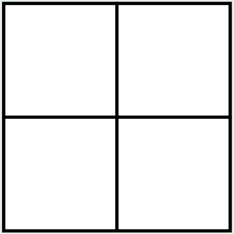
- d cuts for d dimensions
- 2^d new boxes



Splitting: Options

Bisect All Dimensions

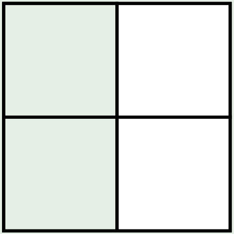
- d cuts for d dimensions
- 2^d new boxes



Splitting: Options

Bisect All Dimensions

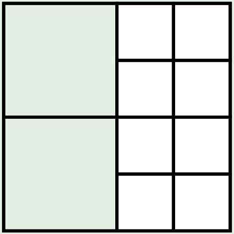
- d cuts for d dimensions
- 2^d new boxes



Splitting: Options

Bisect All Dimensions

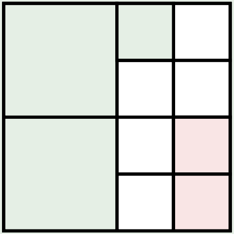
- d cuts for d dimensions
- 2^d new boxes



Splitting: Options

Bisect All Dimensions

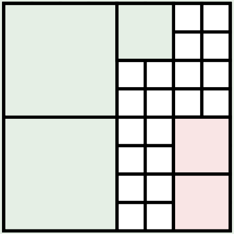
- d cuts for d dimensions
- 2^d new boxes



Splitting: Options

Bisect All Dimensions

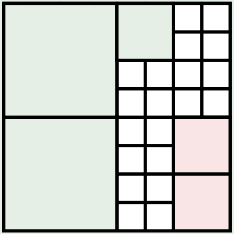
- d cuts for d dimensions
- 2^d new boxes



Splitting: Options

Bisect All Dimensions

- d cuts for d dimensions
- 2^d new boxes

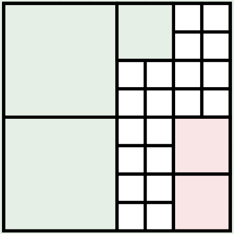


Bisect Single Dimension

Splitting: Options

Bisect All Dimensions

- d cuts for d dimensions
- 2^d new boxes



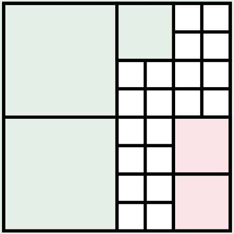
Bisect Single Dimension

- 1 cut for d dimensions

Splitting: Options

Bisect All Dimensions

- d cuts for d dimensions
- 2^d new boxes



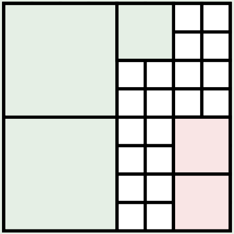
Bisect Single Dimension

- 1 cut for d dimensions
- 2 new boxes

Splitting: Options

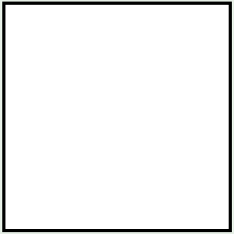
Bisect All Dimensions

- d cuts for d dimensions
- 2^d new boxes



Bisect Single Dimension

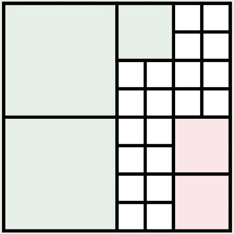
- 1 cut for d dimensions
- 2 new boxes



Splitting: Options

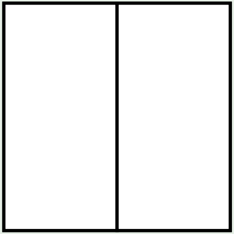
Bisect All Dimensions

- d cuts for d dimensions
- 2^d new boxes



Bisect Single Dimension

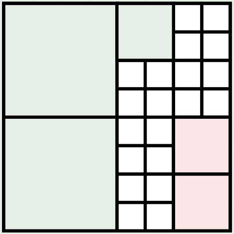
- 1 cut for d dimensions
- 2 new boxes



Splitting: Options

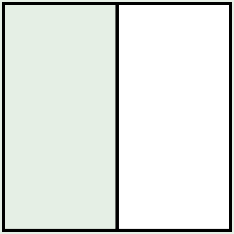
Bisect All Dimensions

- d cuts for d dimensions
- 2^d new boxes



Bisect Single Dimension

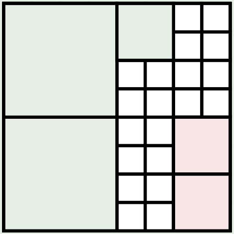
- 1 cut for d dimensions
- 2 new boxes



Splitting: Options

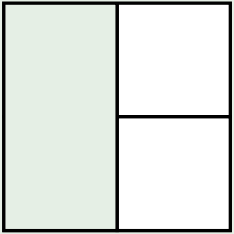
Bisect All Dimensions

- d cuts for d dimensions
- 2^d new boxes



Bisect Single Dimension

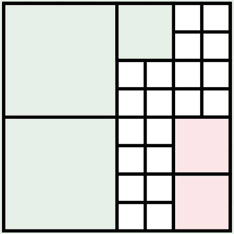
- 1 cut for d dimensions
- 2 new boxes



Splitting: Options

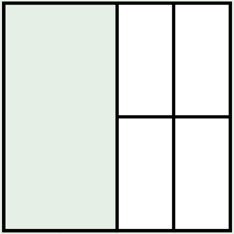
Bisect All Dimensions

- d cuts for d dimensions
- 2^d new boxes



Bisect Single Dimension

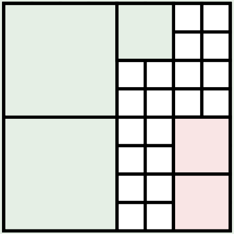
- 1 cut for d dimensions
- 2 new boxes



Splitting: Options

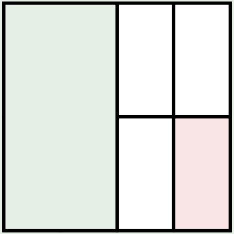
Bisect All Dimensions

- d cuts for d dimensions
- 2^d new boxes



Bisect Single Dimension

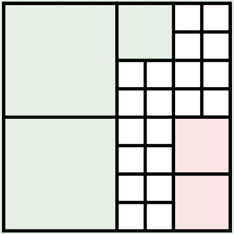
- 1 cut for d dimensions
- 2 new boxes



Splitting: Options

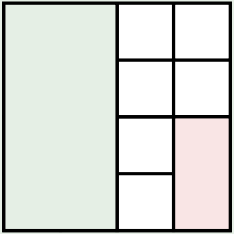
Bisect All Dimensions

- d cuts for d dimensions
- 2^d new boxes



Bisect Single Dimension

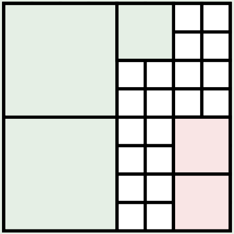
- 1 cut for d dimensions
- 2 new boxes



Splitting: Options

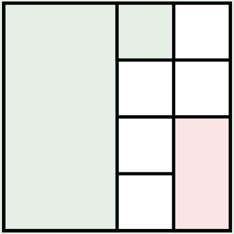
Bisect All Dimensions

- d cuts for d dimensions
- 2^d new boxes



Bisect Single Dimension

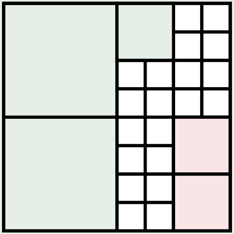
- 1 cut for d dimensions
- 2 new boxes



Splitting: Options

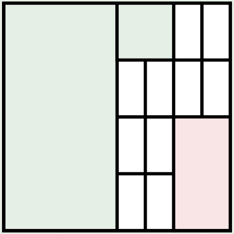
Bisect All Dimensions

- d cuts for d dimensions
- 2^d new boxes



Bisect Single Dimension

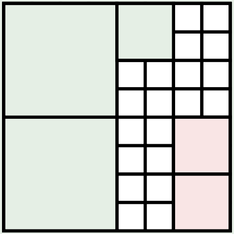
- 1 cut for d dimensions
- 2 new boxes



Splitting: Options

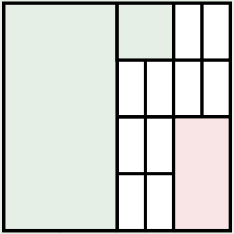
Bisect All Dimensions

- d cuts for d dimensions
- 2^d new boxes



Bisect Single Dimension

- 1 cut for d dimensions
- 2 new boxes

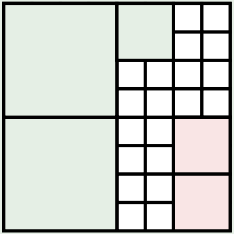


\+ prevent unnecessary splits

Splitting: Options

Bisect All Dimensions

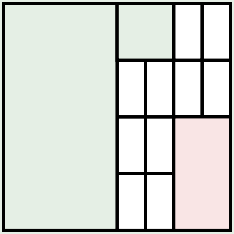
- d cuts for d dimensions
- 2^d new boxes



− introduce unnecessary splits

Bisect Single Dimension

- 1 cut for d dimensions
- 2 new boxes

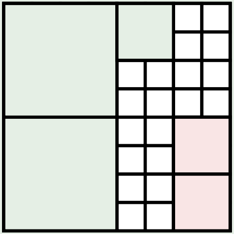


+ prevent unnecessary splits

Splitting: Options

Bisect All Dimensions

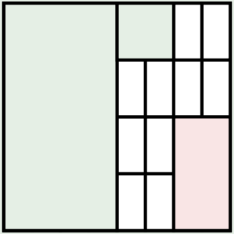
- d cuts for d dimensions
- 2^d new boxes



– introduce unnecessary splits

Bisect Single Dimension

- 1 cut for d dimensions
- 2 new boxes



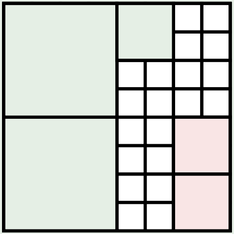
\+ prevent unnecessary splits

– introduce unnecessary solves

Splitting: Options

Bisect All Dimensions

- d cuts for d dimensions
- 2^d new boxes

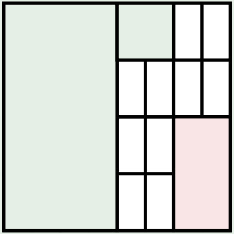


− introduce unnecessary splits

+ prevent unnecessary solves

Bisect Single Dimension

- 1 cut for d dimensions
- 2 new boxes



+ prevent unnecessary splits

− introduce unnecessary solves

Incremental Solving

Idea

Incremental Solving

Idea

- context: deduced logical implications from solving a formula

Incremental Solving

Idea

- context: deduced logical implications from solving a formula
- context of a box B still valid for its child boxes B_c, no need to create new context

Incremental Solving

Idea

- context: deduced logical implications from solving a formula
- context of a box B still valid for its child boxes B_c, no need to create new context

$\Rightarrow$ reuse this context

Incremental Solving

Idea

- context: deduced logical implications from solving a formula
- context of a box B still valid for its child boxes B_c, no need to create new context

$\Rightarrow$ reuse this context

Example

Incremental Solving

Idea

- context: deduced logical implications from solving a formula
- context of a box B still valid for its child boxes B_c, no need to create new context

$\Rightarrow$ reuse this context

Example

- $B = [-1, 1] \times [-1, 1]$

Incremental Solving

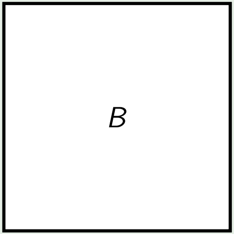
Idea

- context: deduced logical implications from solving a formula
- context of a box B still valid for its child boxes B_c, no need to create new context

$\Rightarrow$ reuse this context

Example

- $B = [-1, 1] \times [-1, 1]$



Incremental Solving

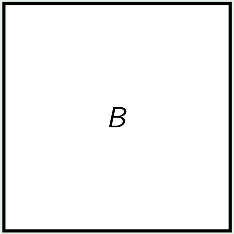
Idea

- context: deduced logical implications from solving a formula
- context of a box B still valid for its child boxes B_c, no need to create new context

$\Rightarrow$ reuse this context

Example

- $B = [-1, 1] \times [-1, 1]$
- check: $B(x) \wedge \varphi(x)$



Incremental Solving

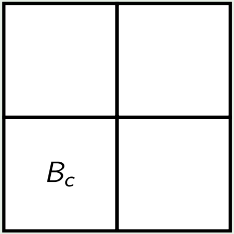
Idea

- context: deduced logical implications from solving a formula
- context of a box B still valid for its child boxes B_c, no need to create new context

$\Rightarrow$ reuse this context

Example

- $B = [-1, 1] \times [-1, 1]$
- check: $B(x) \wedge \varphi(x)$
- $B_c = [-1, 0] \times [-1, 0]$



Incremental Solving

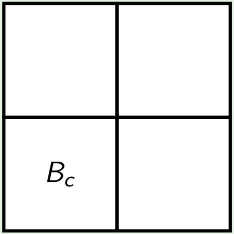
Idea

- context: deduced logical implications from solving a formula
- context of a box B still valid for its child boxes B_c, no need to create new context

$\Rightarrow$ reuse this context

Example

- $B = [-1, 1] \times [-1, 1]$
- check: $B(x) \wedge \varphi(x)$
- $B_c = [-1, 0] \times [-1, 0]$
- check: $B_c(x) \wedge \varphi(x)$



Incremental Solving

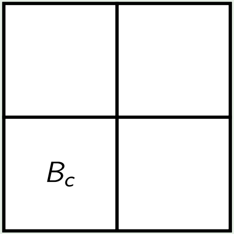
Idea

- context: deduced logical implications from solving a formula
- context of a box B still valid for its child boxes B_c, no need to create new context

$\Rightarrow$ reuse this context

Example

- $B = [-1, 1] \times [-1, 1]$
- check: $B(x) \land \varphi(x)$
- $B_c = [-1, 0] \times [-1, 0]$, thus $B_c \subseteq B$
- check: $B_c(x) \land \varphi(x)$



Incremental Solving

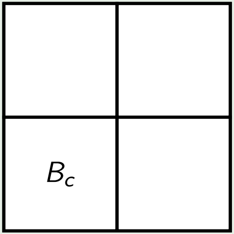
Idea

- context: deduced logical implications from solving a formula
- context of a box B still valid for its child boxes B_c, no need to create new context

$\Rightarrow$ reuse this context

Example

- $B = [-1, 1] \times [-1, 1]$
- check: $B(x) \land \varphi(x)$
- $B_c = [-1, 0] \times [-1, 0]$, thus $B_c \subseteq B$, thus $B_c(x) \to B(x)$
- check: $B_c(x) \land \varphi(x)$



Incremental Solving

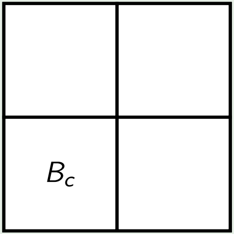
Idea

- context: deduced logical implications from solving a formula
- context of a box B still valid for its child boxes B_c, no need to create new context

$\Rightarrow$ reuse this context

Example

- $B = [-1, 1] \times [-1, 1]$
- check: $B(x) \wedge \varphi(x)$
- $B_c = [-1, 0] \times [-1, 0]$, thus $B_c \subseteq B$, thus $B_c(x) \to B(x)$
- check: $B_c(x) \wedge \varphi(x)$ $\equiv B_c(x) \wedge (B(x) \wedge \varphi(x))$



Incremental Solving

Idea

- context: deduced logical implications from solving a formula
- context of a box B still valid for its child boxes B_c, no need to create new context

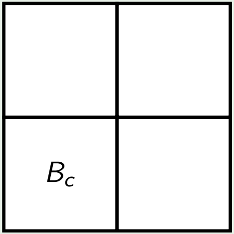
$\Rightarrow$ reuse this context

Implementation

- single context for a box and its children

Example

- $B = [-1, 1] \times [-1, 1]$
- check: $B(x) \wedge \varphi(x)$
- $B_c = [-1, 0] \times [-1, 0]$, thus $B_c \subseteq B$, thus $B_c(x) \rightarrow B(x)$
- check: $B_c(x) \wedge \varphi(x)$ $\equiv B_c(x) \wedge (B(x) \wedge \varphi(x))$



GUI Projection

GUI Projection

Graphical User Interface

- The GUI is able to depict only two dimensions.

GUI Projection

Graphical User Interface

- The GUI is able to depict only two dimensions.

Problem

- The algorithm can handle more than two dimensions.

GUI Projection

Graphical User Interface

- The GUI is able to depict only two dimensions.

Problem

- The algorithm can handle more than two dimensions.

Solution

- Existential Projection!

GUI Projection

Graphical User Interface

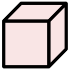
- The GUI is able to depict only two dimensions.

Problem

- The algorithm can handle more than two dimensions.

Solution

- Existential Projection!



GUI Projection

Graphical User Interface

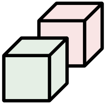
- The GUI is able to depict only two dimensions.

Problem

- The algorithm can handle more than two dimensions.

Solution

- Existential Projection!



GUI Projection

Graphical User Interface

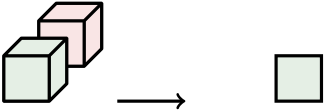
- The GUI is able to depict only two dimensions.

Problem

- The algorithm can handle more than two dimensions.

Solution

- Existential Projection!



GUI Projection

Graphical User Interface

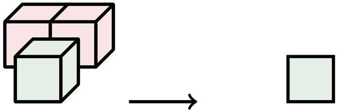
- The GUI is able to depict only two dimensions.

Problem

- The algorithm can handle more than two dimensions.

Solution

- Existential Projection!



GUI Projection

Graphical User Interface

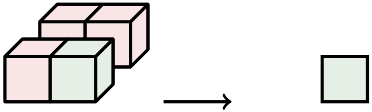
- The GUI is able to depict only two dimensions.

Problem

- The algorithm can handle more than two dimensions.

Solution

- Existential Projection!



GUI Projection

Graphical User Interface

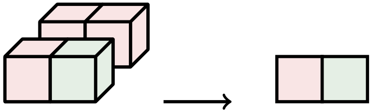
- The GUI is able to depict only two dimensions.

Problem

- The algorithm can handle more than two dimensions.

Solution

- Existential Projection!



GUI Projection

Graphical User Interface

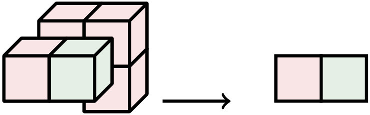
- The GUI is able to depict only two dimensions.

Problem

- The algorithm can handle more than two dimensions.

Solution

- Existential Projection!



GUI Projection

Graphical User Interface

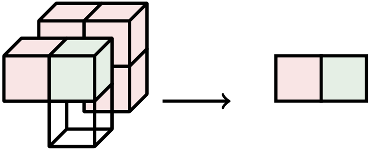
- The GUI is able to depict only two dimensions.

Problem

- The algorithm can handle more than two dimensions.

Solution

- Existential Projection!



GUI Projection

Graphical User Interface

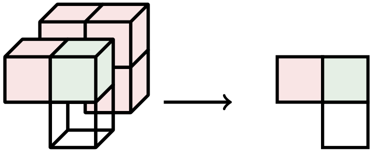
- The GUI is able to depict only two dimensions.

Problem

- The algorithm can handle more than two dimensions.

Solution

- Existential Projection!



GUI Projection

Graphical User Interface

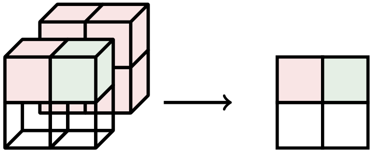
- The GUI is able to depict only two dimensions.

Problem

- The algorithm can handle more than two dimensions.

Solution

- Existential Projection!



GUI Projection

Graphical User Interface

- The GUI is able to depict only two dimensions.

Problem

- The algorithm can handle more than two dimensions.

Solution

- Existential Projection!

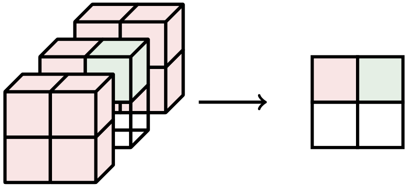
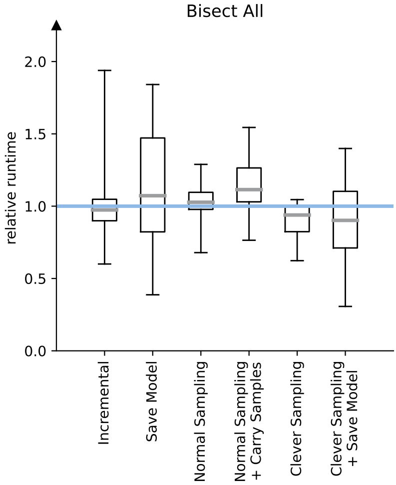


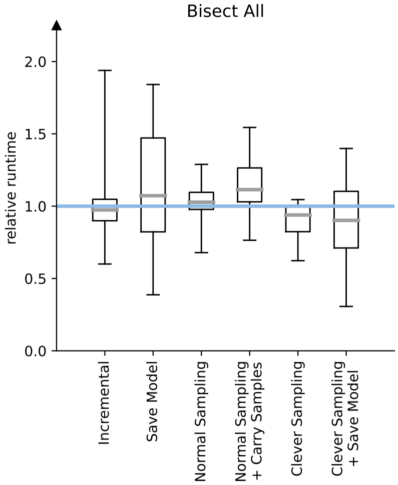
Table of Contents

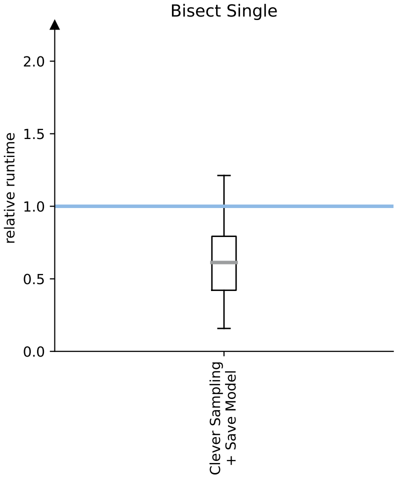
Speed-Up Factors

Speed-Up Factors



Speed-Up Factors





Solver Call Prevention

Solver Call Prevention

	Bisect All		Bisect Single	
	Preventable	Prevented	Preventable	Prevented
Base Algorithm	0.55	0.00	0.83	0.00
Incremental	0.55	0.00	-	-
Save Model	0.55	0.17	-	-
Normal Sampling	0.55	0.29	-	-
+ Carry Samples	0.55	0.31	-	-
Clever Sampling	0.55	0.29	-	-
+ Save Model	0.55	0.39	0.83	0.72

Splitting Heuristic Comparison

Splitting Heuristic Comparison

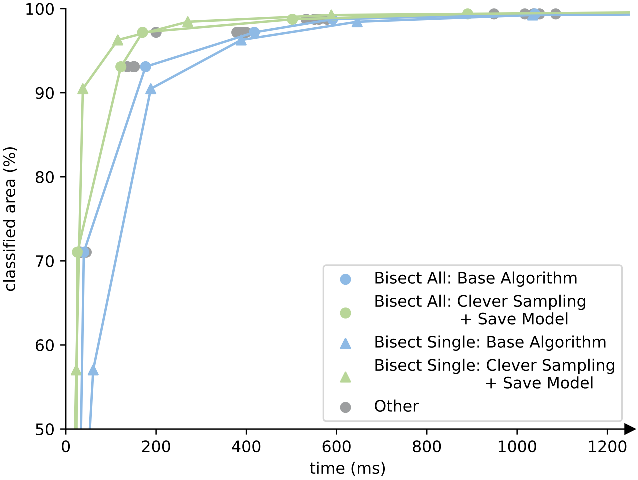


Table of Contents

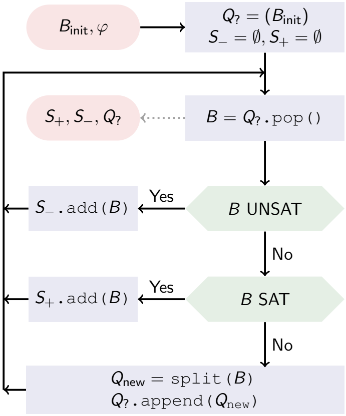
Conclusion

Conclusion

Topic

Topic

- parameter synthesis

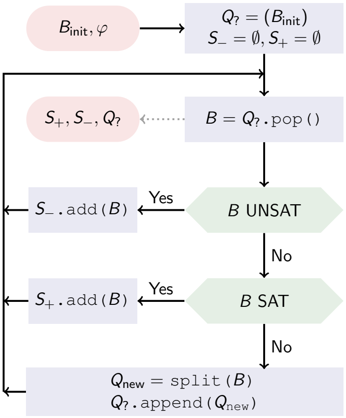


Conclusion

Topic

- parameter synthesis

Performance



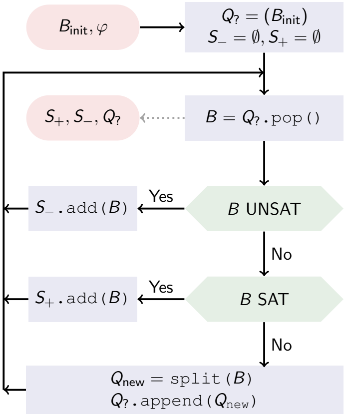
Conclusion

Topic

- parameter synthesis

Performance

- good compared to PaSyPy



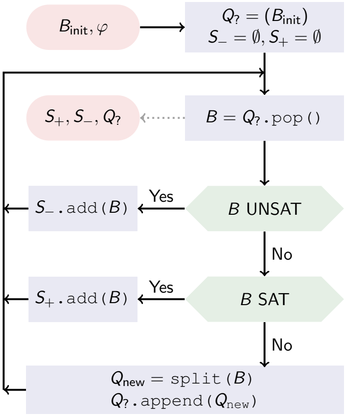
Conclusion

Topic

- parameter synthesis

Performance

- good compared to PaSyPy
- dependent on Z3 performance



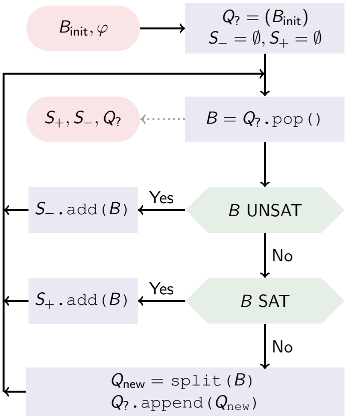
Conclusion

Topic

- parameter synthesis

Performance

- good compared to PaSyPy
- dependent on Z3 performance
- implemented heuristics is very benchmark dependent



Conclusion

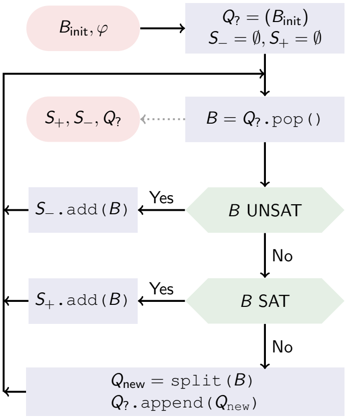
Topic

- parameter synthesis

Performance

- good compared to PaSyPy
- dependent on Z3 performance
- implemented heuristics is very benchmark dependent

Open Questions & Future Work



Conclusion

Topic

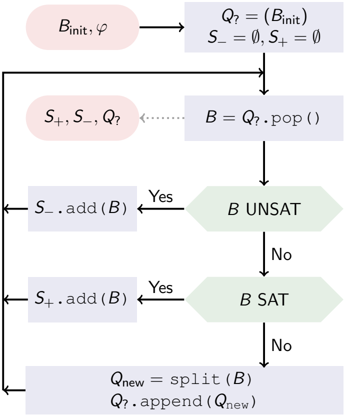
- parameter synthesis

Performance

- good compared to PaSyPy
- dependent on Z3 performance
- implemented heuristics is very benchmark dependent

Open Questions & Future Work

- reason for low correlation of solving time and number of solver calls



Conclusion

Topic

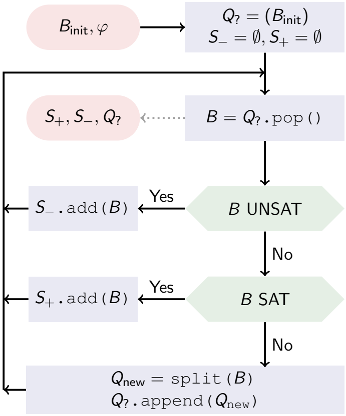
- parameter synthesis

Performance

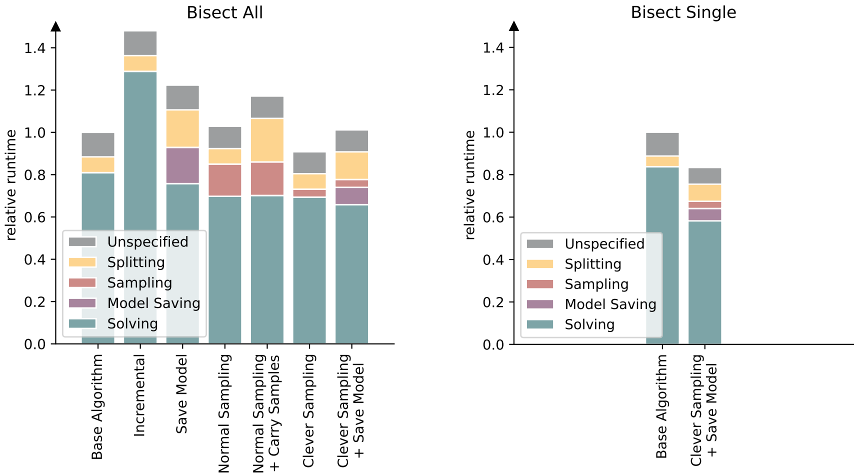
- good compared to PaSyPy
- dependent on Z3 performance
- implemented heuristics is very benchmark dependent

Open Questions & Future Work

- reason for low correlation of solving time and number of solver calls
- advanced splitting heuristics



Experimental Evaluation: Time Distribution



Evaluation: Average Speed-Up Factors

Bisect All

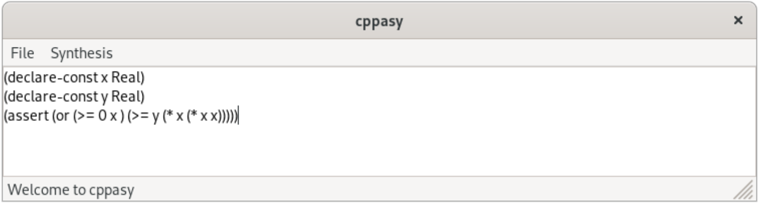
	4^{1910}	5^{1910}	6^{1910}	7^{1843}	8^{1319}	9^{922}	10^{732}
Clever Sampling	0.85	0.96	0.91	0.98	0.91	1.04	0.95
Base Algorithm	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Clever Sampling +Save Model	1.57	1.33	1.01	1.03	0.93	1.04	0.91
Normal Sampling	1.02	1.01	1.03	1.03	1.05	1.10	1.10
Normal Sampling +Carry Samples	1.12	1.12	1.17	1.15	1.17	1.23	1.23
Save Model	1.92	1.39	1.22	1.19	1.16	1.13	1.09
Incremental	1.39	-	1.48	-	1.1	-	1.07

Evaluation: Average Speed-Up Factors

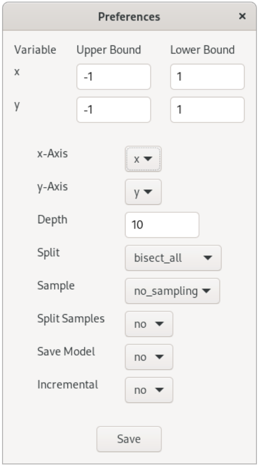
Bisect Single

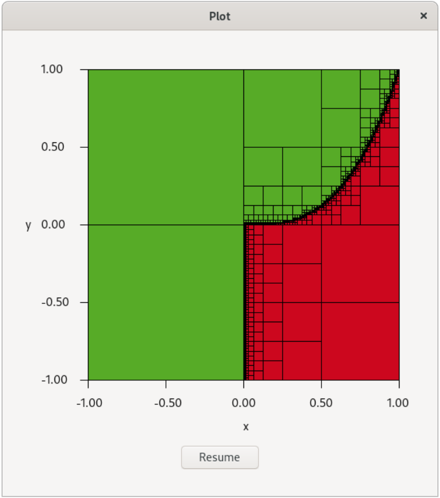
	4^{1910}	5^{1910}	6^{1910}	7^{1578}	8^{1008}	9^{763}	10^{568}
Clever Sampling +Save Model	0.79	0.77	0.83	0.71	0.64	0.61	0.59
Base Algorithm	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Implementation: GUI



Implementation: GUI





Implementation: CLI

```
$ ./build/cli --help
Allowed options:
  -h [ --help ]          produce help message
  --boundaries-file arg Text file containing a list of all variables and their
                         boundaries. The file should contain lines of the from
                         '<variable-name> <lower-bound> <upper-bound>'.
  --default-boundaries arg (=10)
                         Set default 'radius' of the inital orthotope.
  --splitting-heuristic arg (=bisect_all)
                         Select a splitting heuristic. Options are 'bisect_all'
                         and 'bisect_single'
  --sampling-heuristic arg (=no_sampling)
                         Select a sampling heuristic. Options are 'no_sampling',
                         'center', and clever'
  --max-depth arg (=10) maximal depth.
  --save-model           Save models found by solver. Only useful if
                         'split-samples' enabled.
  --incremental          Enable incremental solving.
  --split-samples        Also carry samples when splitting orthotopes.
  --splits-needed        Returns true if splits are needed to process this
                         formula.
  --print-orthotopes     Prints all (SAFE, UNSAFE and UNKNOWN) resulting
                         orthotopes.
```