Reactivity-aware Discrete Controller Synthesis for Logico-numerical Reactive Programs

Nicolas BERTHIER Hervé MARCHAND

SUMO Team
Inria Rennes — Bretagne Atlantique

SYNCHRON' 14

December 1, 2014

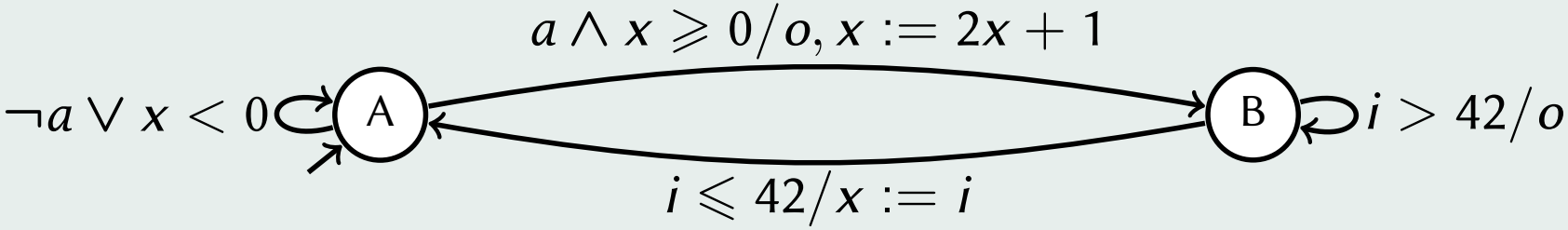
Outline

- ASTS Model of Logico-numerical Reactive Programs
- Safety Control Problem for ASTSs
- Discrete Controller Synthesis for ASTSs
- Some Limitations & Solutions
- ReaX
- Conclusions

Outline

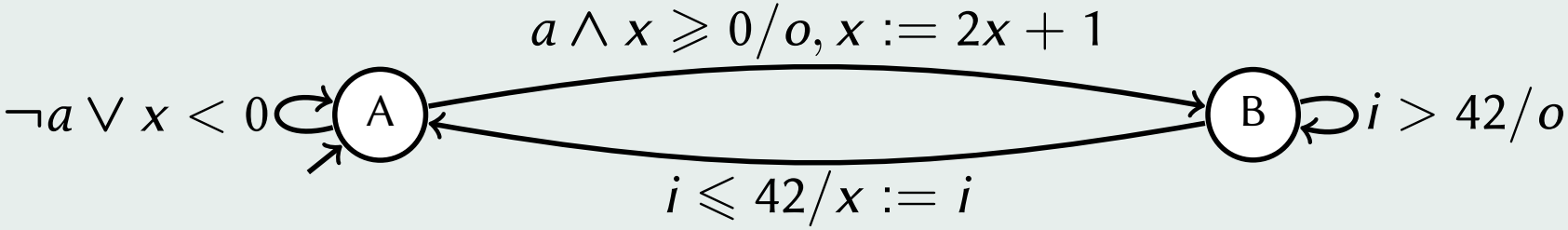
Example Logico-numerical Program & ASTS Model

Example Automaton



Example Logico-numerical Program & ASTS Model

Example Automaton



Arithmetic Symbolic Transition System Model (ASTS)

$X = \langle \xi, x, o \rangle, I = \langle a, i \rangle$ $\qquad \mathcal{D}_X = \{\mathrm{A}, \mathrm{B}\} \times \mathbb{Z} \times \mathbb{B}, \mathcal{D}_I = \mathbb{B} \times \mathbb{Z}$

$$T = \begin{cases} \xi' = & \mathrm{B} & \text{if } (\xi = \mathrm{A} \wedge a \wedge x \geqslant 0), \\ & \mathrm{A} & \text{if } (\xi = \mathrm{B} \wedge i > 42), \xi \text{ othewise} \\ x' = & 2x + 1 & \text{if } (\xi = \mathrm{A} \wedge a \wedge x \geqslant 0), \\ & i & \text{if } (\xi = \mathrm{B} \wedge i \leqslant 42), x \text{ othewise} \\ o' = & (\xi = \mathrm{A} \wedge a \wedge x \geqslant 0) \vee (\xi = \mathrm{B} \wedge i > 42) \end{cases}$$

$A(\langle \xi, x, o, a, i \rangle) = (\xi = \mathrm{B} \wedge 3x + 2i \leqslant 41 \wedge a)$
$\Theta_0(\langle \xi, x, o \rangle) = (\xi = \mathrm{A} \wedge x = 0 \wedge \neg o)$

Outline

Safety Control Problem

Initiated by Ramadge and Wonham 1989[1]

Definition (Invariant for an ASTS)

Given an ASTS $S = \langle X, I, T, A, \Theta_0 \rangle$, a Predicate Φ over X is an *Invariant* of S (Noted $S \models \Phi$) iff All Reachable States of S Satisfy Φ

[1] Peter J. G. Ramadge and W. Murray Wonham. "The control of discrete event systems". In: *Proceedings of the IEEE* 77.1 (Jan. 1989), pp. 81–98.

Safety Control Problem

Initiated by Ramadge and Wonham 1989[1]

Definition (Invariant for an ASTS)

Given an ASTS $S = \langle X, I, T, A, \Theta_0 \rangle$, a Predicate Φ over X is an *Invariant* of S (Noted $S \models \Phi$) iff All Reachable States of S Satisfy Φ

Controller Synthesis Problem for Invariant Enforcement in ASTSs

Given and ASTS $S = \langle X, U \uplus C, T, A, \Theta_0 \rangle$ where:

- U ← *Non-controllable* Input Variables
- C ← *Controllable* Input Variables

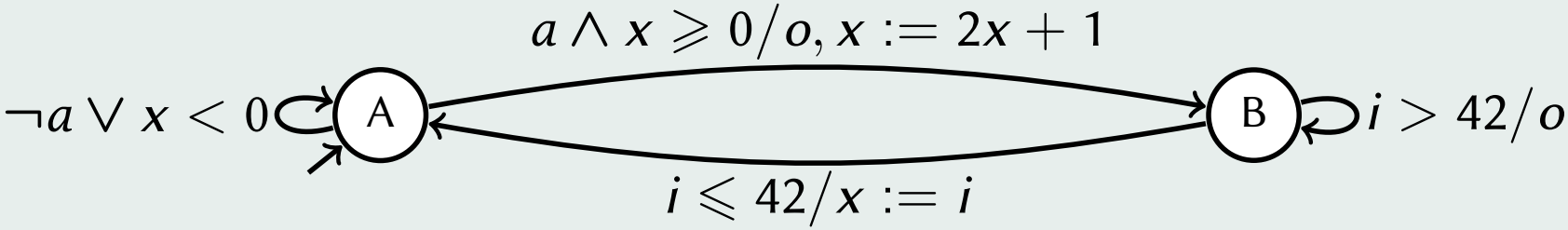
and an Invariant Φ over X, ← Not Satisfied *a priori*
Compute a Predicate K_Φ such that:

$$S' = \langle X, U \uplus C, T, K_\Phi, \Theta_0 \rangle \models \Phi \qquad \text{and} \qquad \forall v, K_\Phi(v) \Rightarrow A(v)$$

[1] Peter J. G. Ramadge and W. Murray Wonham. "The control of discrete event systems". In: *Proceedings of the IEEE* 77.1 (Jan. 1989), pp. 81–98.

Example Control Problem for an ASTS

Example Automaton



Controllable ASTS

$X = \langle \xi, x, o \rangle$, $U = \langle i \rangle$, $C = \langle a \rangle$ $\quad \mathcal{D}_X = \{\mathrm{A}, \mathrm{B}\} \times \mathbb{Z} \times \mathbb{B}$, $\mathcal{D}_U = \mathbb{Z}$, $\mathcal{D}_C = \mathbb{B}$

$$T = \begin{cases} \xi' = & \mathrm{B} & \text{if } (\xi = \mathrm{A} \wedge a \wedge x \geqslant 0), \\ & \mathrm{A} & \text{if } (\xi = \mathrm{B} \wedge i > 42), \xi \text{ othewise} \\ x' = & 2x + 1 & \text{if } (\xi = \mathrm{A} \wedge a \wedge x \geqslant 0), \\ & i & \text{if } (\xi = \mathrm{B} \wedge i \leqslant 42), x \text{ othewise} \\ o' = & (\xi = \mathrm{A} \wedge a \wedge x \geqslant 0) \vee (\xi = \mathrm{B} \wedge i > 42) \end{cases}$$

$A(\langle \xi, x, o, a, i \rangle) = \mathrm{tt}$

$\Theta_0(\langle \xi, x, o \rangle) = (\xi = \mathrm{A} \wedge x = 0 \wedge \neg o)$

$\Phi(\langle \xi, x, o \rangle) = (\xi = \mathrm{B} \Rightarrow x \leqslant 10)$

Outline

- ASTS Model of Logico-numerical Reactive Programs
- Safety Control Problem for ASTSs
- Discrete Controller Synthesis for ASTSs
  - Symbolic Algorithm
  - Finite Case
  - Infinite Case
- Some Limitations & Solutions
- ReaX
- Conclusions

Symbolic Computations / Binary Decision Diagrams (BDDs)

Representing Finite Sets with Characteristic Functions

- Boolean Variables $V = \langle v_1, \ldots, v_n \rangle \rightsquigarrow$ Universe $\mathcal{D}_V = \mathbb{B}^n$
- Boolean Function $P\colon \mathbb{B}^n \to \mathbb{B} \rightsquigarrow$ Set $\mathcal{P} \subseteq \mathcal{D}_V$ $\qquad (\mathcal{P} \in \wp(\mathcal{D}_V))$
- Notations:

$$\mathsf{ff} \rightsquigarrow \varnothing \qquad \mathsf{tt} \rightsquigarrow \mathcal{D}_V \qquad \neg P \rightsquigarrow \mathcal{D}_V \setminus \mathcal{P}$$

$$P \wedge Q \rightsquigarrow \mathcal{P} \cap \mathcal{Q} \qquad P \vee Q \rightsquigarrow \mathcal{P} \cup \mathcal{Q} \qquad \exists v \in \mathcal{D}_v, P \qquad \forall v \in \mathcal{D}_v, P$$

Symbolic Computations / Binary Decision Diagrams (BDDs)

Representing Finite Sets with Characteristic Functions

- Boolean Variables $V = \langle v_1, \ldots, v_n \rangle \rightsquigarrow$ Universe $\mathcal{D}_V = \mathbb{B}^n$
- Boolean Function $P\colon \mathbb{B}^n \to \mathbb{B} \rightsquigarrow$ Set $\mathcal{P} \subseteq \mathcal{D}_V$ $\qquad (\mathcal{P} \in \wp(\mathcal{D}_V))$
- Notations:

$$\mathrm{ff} \rightsquigarrow \varnothing \qquad \mathrm{tt} \rightsquigarrow \mathcal{D}_V \qquad \neg P \rightsquigarrow \mathcal{D}_V \setminus \mathcal{P}$$
$$P \wedge Q \rightsquigarrow \mathcal{P} \cap \mathcal{Q} \qquad P \vee Q \rightsquigarrow \mathcal{P} \cup \mathcal{Q} \qquad \exists v \in \mathcal{D}_v, P \qquad \forall v \in \mathcal{D}_v, P$$

Adding Arithmetic Constraints to Represent Infinite Sets

- Extending Formulas with Interpreted Variables

$$v_{x \leqslant 42} \in \mathbb{B}: \qquad v_{x \leqslant 42} \rightsquigarrow x \leqslant 42 \qquad \neg v_{x \leqslant 42} \rightsquigarrow x > 42$$

- *e.g.*,

$$V = \langle s, x, y \rangle, \mathcal{D}_V = \mathbb{B} \times \mathbb{Z}^2: \qquad s \wedge v_{x+y \geqslant 10} \wedge \neg v_{x<0}$$

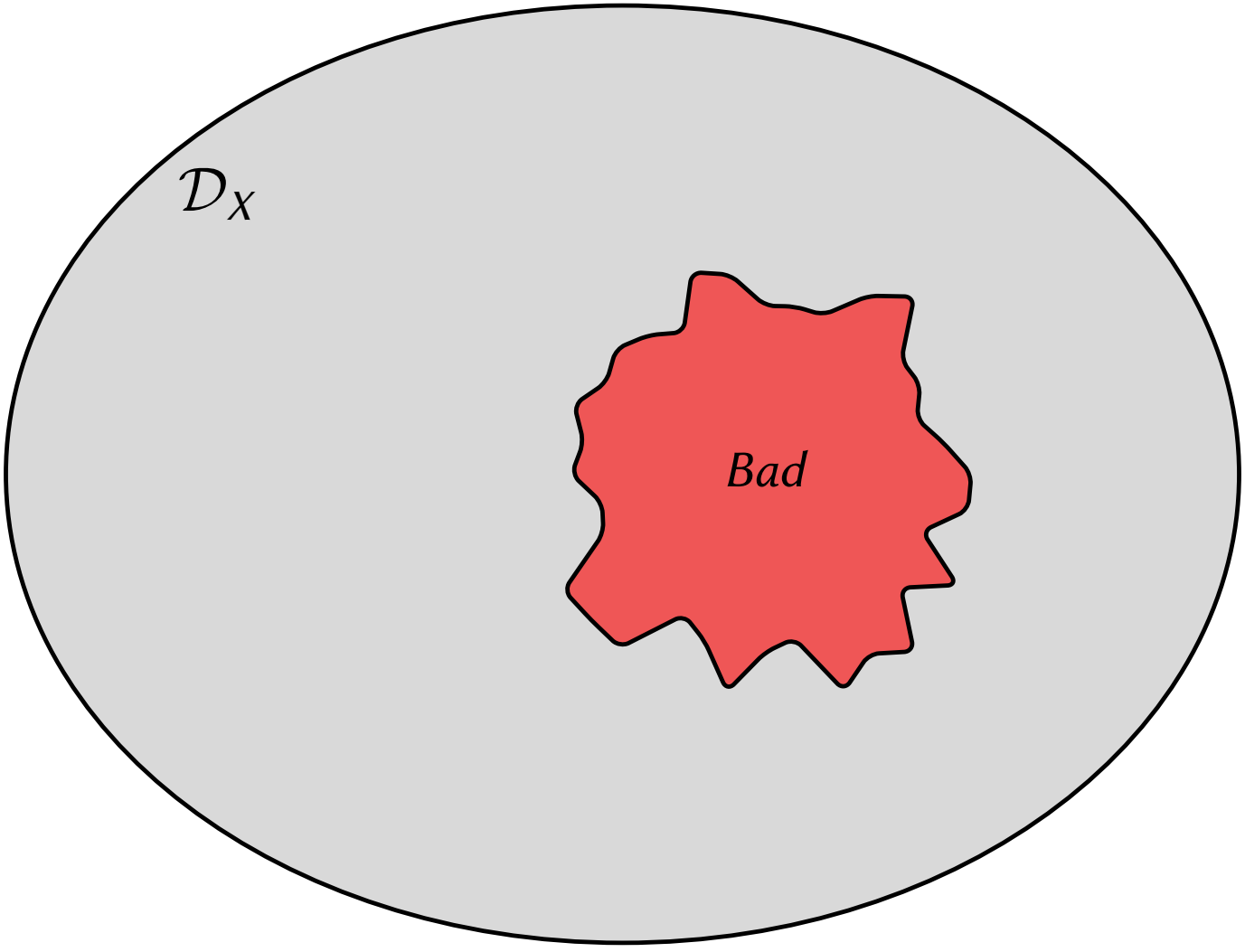
Symbolic Algorithm for Discrete Controller Synthesis

Symbolic Algorithm

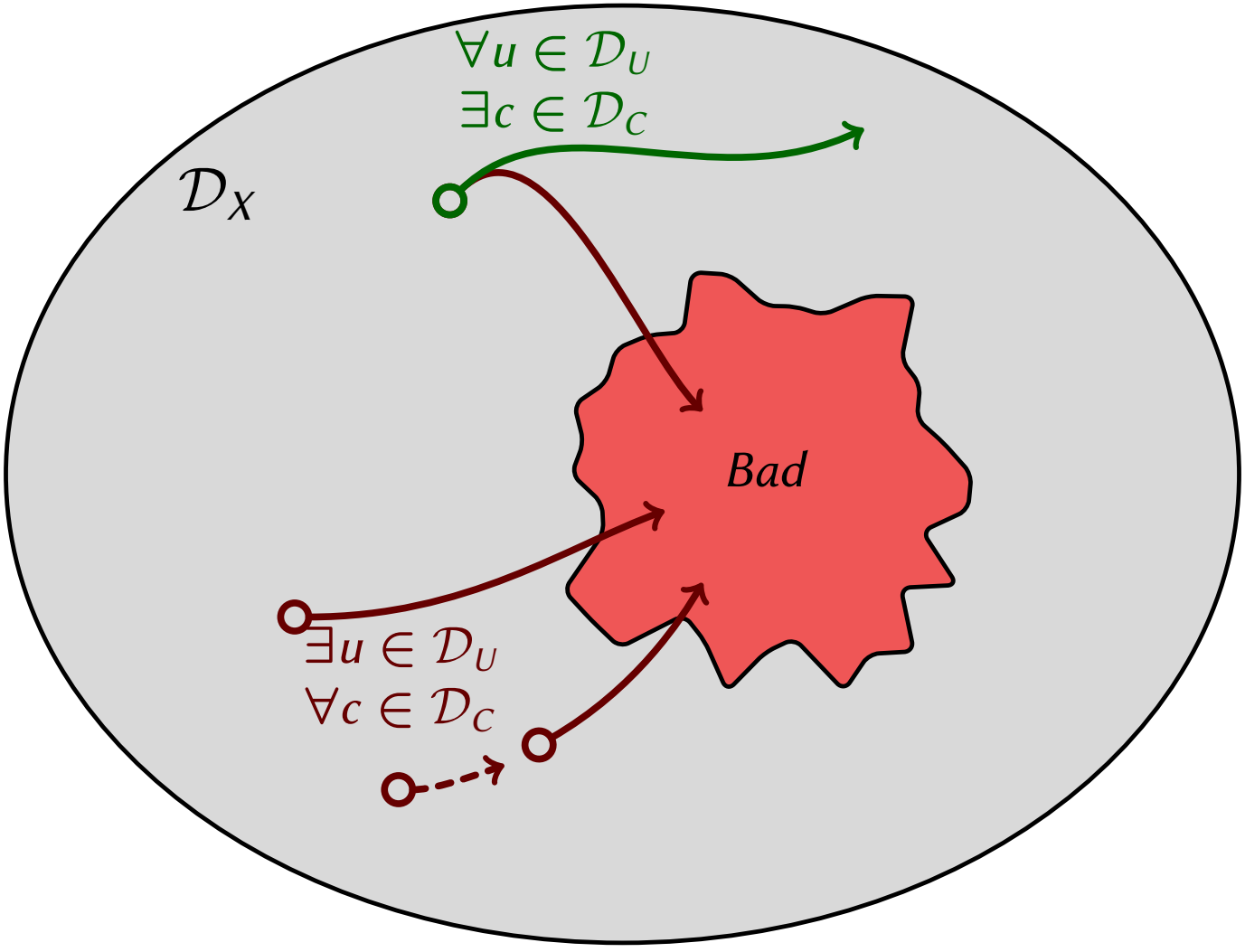
- Let $Bad = \neg\Phi$ $\leftarrow$ States to Avoid
- I_{Bad} = "States *Uncontrollably* Reaching *Bad* " $\leftarrow$ Core Computation
- Success iff $\Theta_0 \wedge I_{Bad} = \mathrm{ff}$ $\leftarrow$ "Initial State(s) $\cap I_{Bad} = \varnothing$"
- $K_\Phi = T^{-1}(\neg I_{Bad}) \wedge A$ $\leftarrow$ Relating States with *Allowed Inputs*

- T^{-1}: "Open" Pre-image
  - Substitution of State Variables According to T

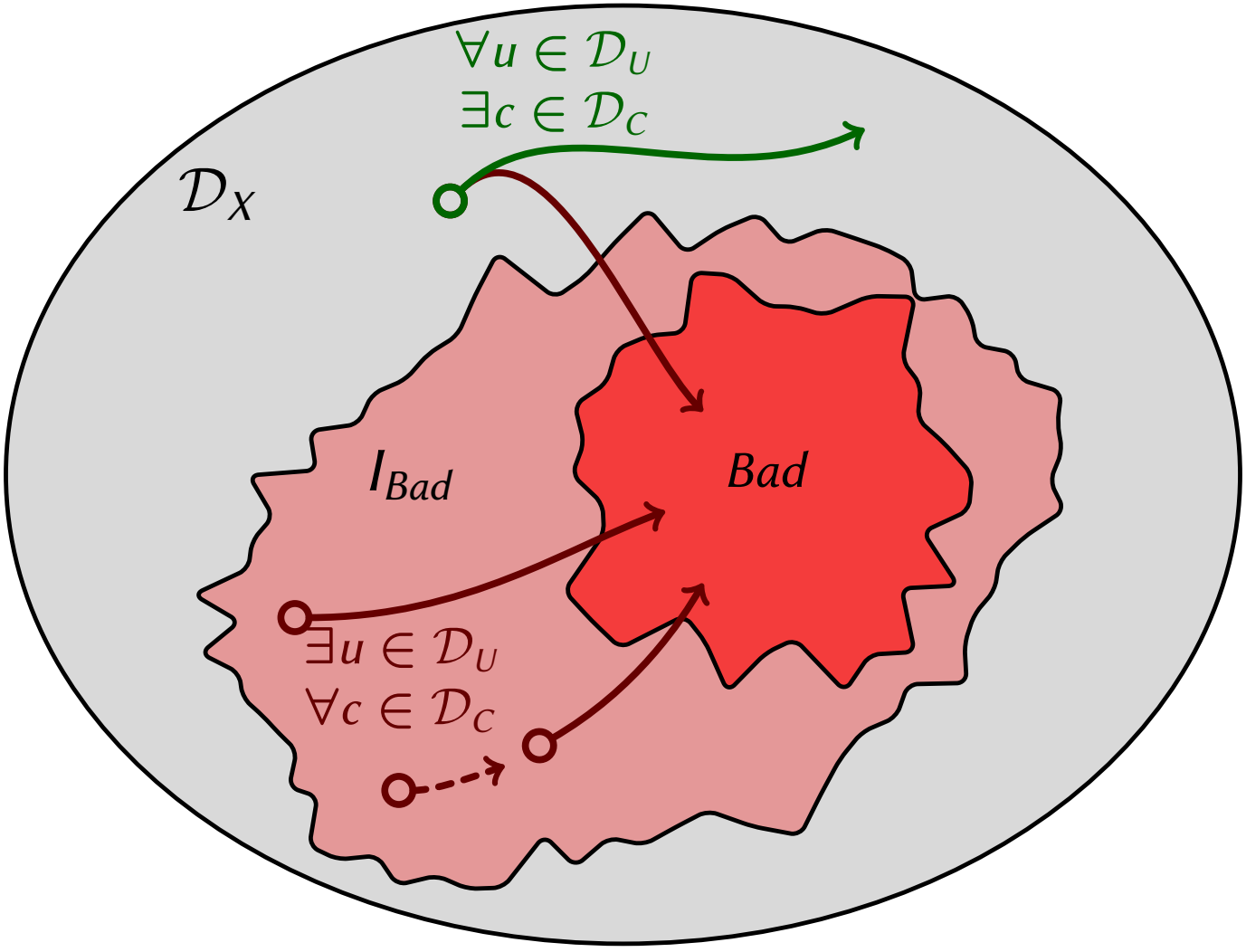
Finite Case: Computing I_{Bad}



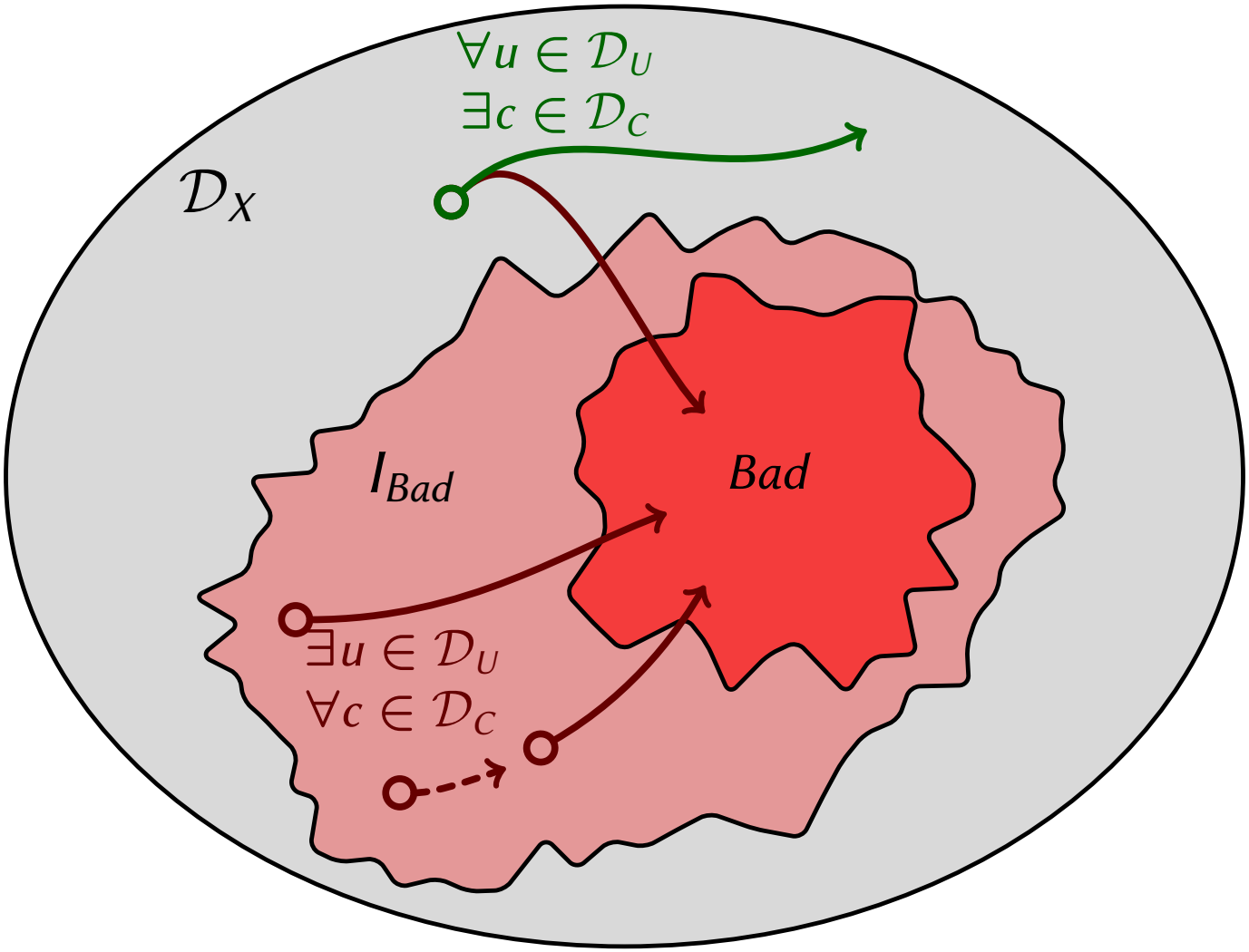
Finite Case: Computing I_{Bad}



Finite Case: Computing I_{Bad}



Finite Case: Computing I_{Bad}



$$I_{Bad} = \text{coreach}_u(Bad) \qquad \text{coreach}_u(B) \stackrel{\text{def}}{=} \text{lfp}\,(\lambda\beta.B \vee \text{pre}_u(\beta))$$

$$\text{pre}_u(B) \stackrel{\text{def}}{=} \exists_U, \forall_C, (T^{-1}(B) \wedge A)$$

Infinite Case: Algorithmic Principle

Infinite Case: Allowing *Numerical* Variables ($e.g., \mathcal{D}_X = \mathbb{B}^n \times \mathbb{Z}^m$)

- Undecidability Problem $\rightsquigarrow$ Over-approximating Solution
  - Using *Abstract Interpretation Techniques* to Compute $I'_{Bad} \supseteq I_{Bad}$

Infinite Case: Algorithmic Principle

Infinite Case: Allowing *Numerical* Variables ($e.g., \mathcal{D}_X = \mathbb{B}^n \times \mathbb{Z}^m$)

- Undecidability Problem $\rightsquigarrow$ Over-approximating Solution
  - Using *Abstract Interpretation Techniques* to Compute $I'_{Bad} \supseteq I_{Bad}$

Abstract Interpretation Requirements

- $\langle \Lambda, \sqsubseteq, \sqcup, \sqcap, \top, \bot \rangle$, α and γ such that: $\wp(\mathcal{D}_X) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \Lambda$
  - $\wp(\mathcal{D}_X)$ $\leftarrow$ Concrete Domain (Sets of States)
  - Λ $\leftarrow$ Abstract Domain (Approximate Representation of Sets of States)
  - $\alpha : \wp(\mathcal{D}_X) \rightarrow \Lambda$ $\leftarrow$ Abstraction Function
  - $\gamma : \Lambda \rightarrow \wp(\mathcal{D}_X)$ $\leftarrow$ Concretization Function
- $T^{\sharp -1}$ $\leftarrow$ Abstract Pre-image
- $\exists^{\sharp}_Y, \forall^{\sharp}_Z$ $\leftarrow$ Eliminations
- $\boldsymbol{\nabla}$ $\leftarrow$ Widening Operator, Forcing Convergence
  - $\ell_1 \boldsymbol{\nabla} \ell_2$ Defined iff $\ell_1 \sqsubseteq \ell_2$

Infinite Case: Logico-numerical Abstract Domains

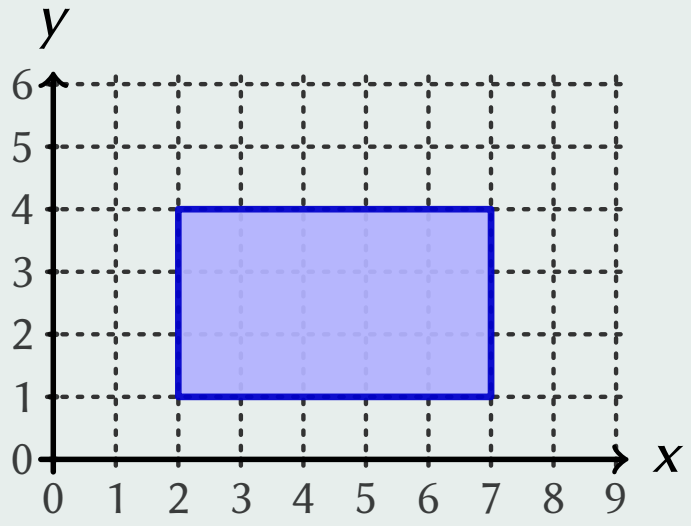
Over-approximating Numerical Spaces ($e.g., \mathcal{D}_X = \mathbb{Z}^n \times \mathbb{R}^m$)

Numerical Abstract Domains

- Provide α, γ and $\mathcal{N}$ such that $\wp(\mathbb{Z}^n \times \mathbb{R}^m) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \mathcal{N}$

e.g.,

- Boxes



Infinite Case: Logico-numerical Abstract Domains

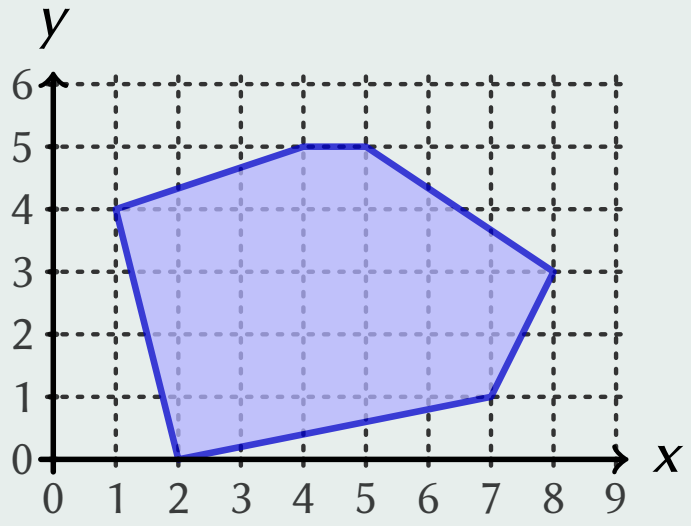
Over-approximating Numerical Spaces (*e.g.*, $\mathcal{D}_X = \mathbb{Z}^n \times \mathbb{R}^m$)

Numerical Abstract Domains

- Provide α, γ and $\mathcal{N}$ such that $\wp(\mathbb{Z}^n \times \mathbb{R}^m) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \mathcal{N}$

e.g.,

- Boxes
- Convex Polyhedra



Infinite Case: Logico-numerical Abstract Domains

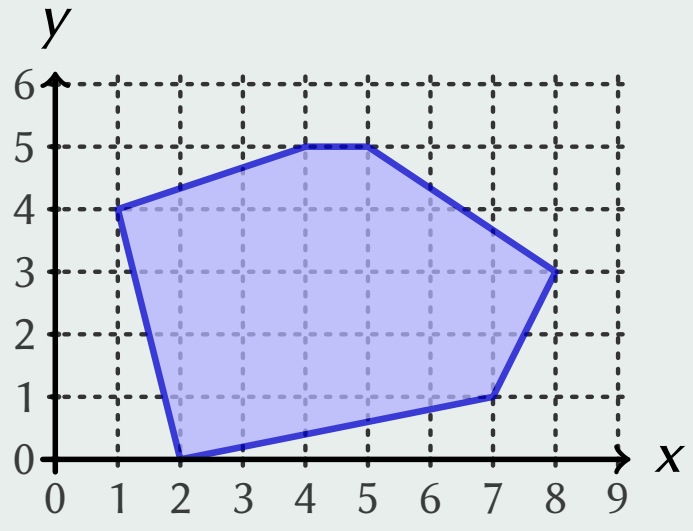
Over-approximating Numerical Spaces ($e.g., \mathcal{D}_X = \mathbb{Z}^n \times \mathbb{R}^m$)

Numerical Abstract Domains

- Provide α, γ and $\mathcal{N}$ such that $\wp(\mathbb{Z}^n \times \mathbb{R}^m) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \mathcal{N}$

e.g.,

- Boxes
- Convex Polyhedra

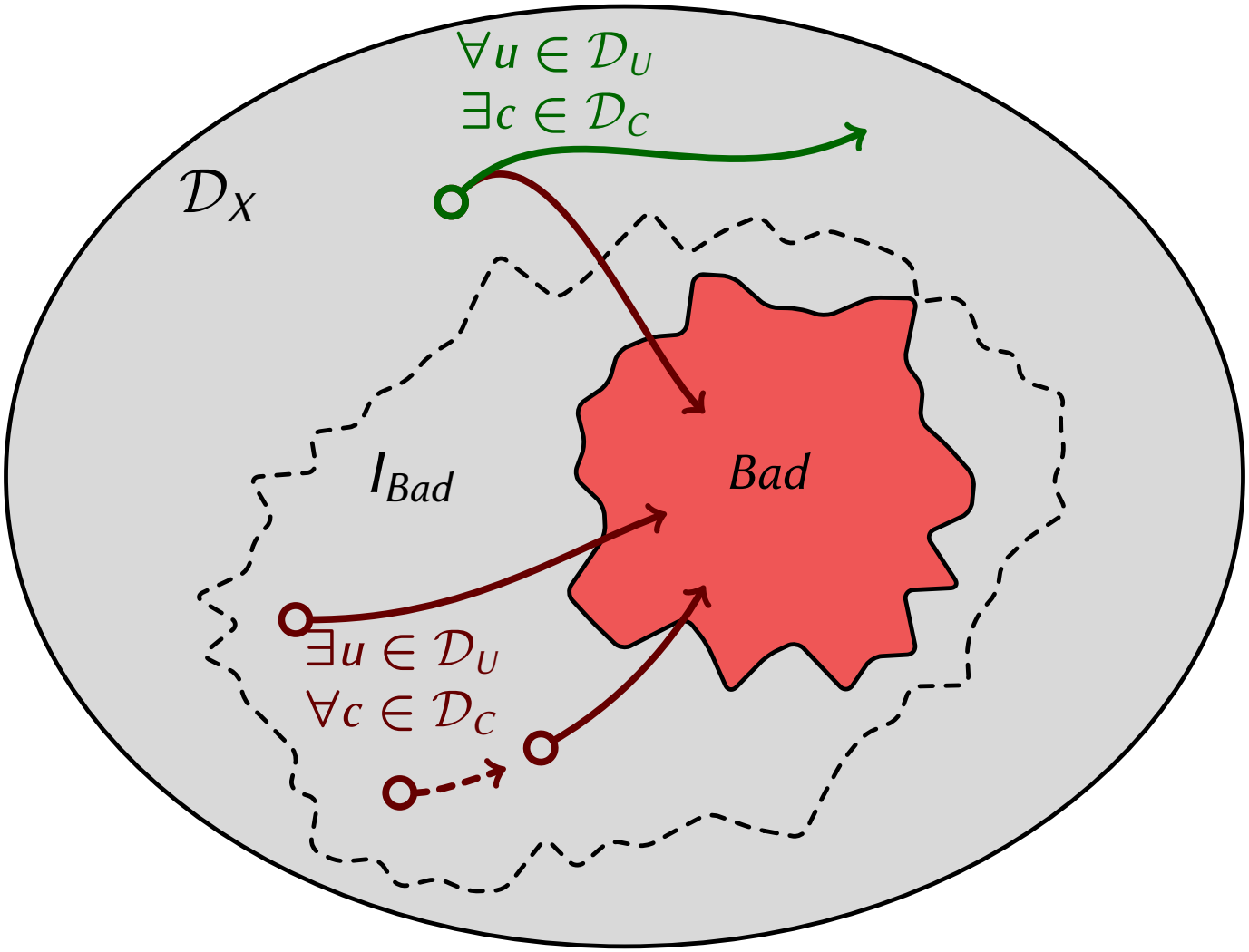


Over-approximating Logico-numerical Spaces ($e.g., \mathcal{D}_X = \mathbb{B}^n \times \mathbb{Z}^m \times \mathbb{R}^o$)

Combining a Boolean Domain and a Numerical Abstract Domain $\mathcal{N}$, *e.g.,*

- *Power*: $\wp(\mathbb{B}^n \times \mathbb{Z}^m \times \mathbb{R}^o) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \mathbb{B}^n \to \mathcal{N}$ $(= \mathcal{N}^{\mathbb{B}^n})$
  - $\rightsquigarrow$ BDDs Associated with Numerical Abstract Values

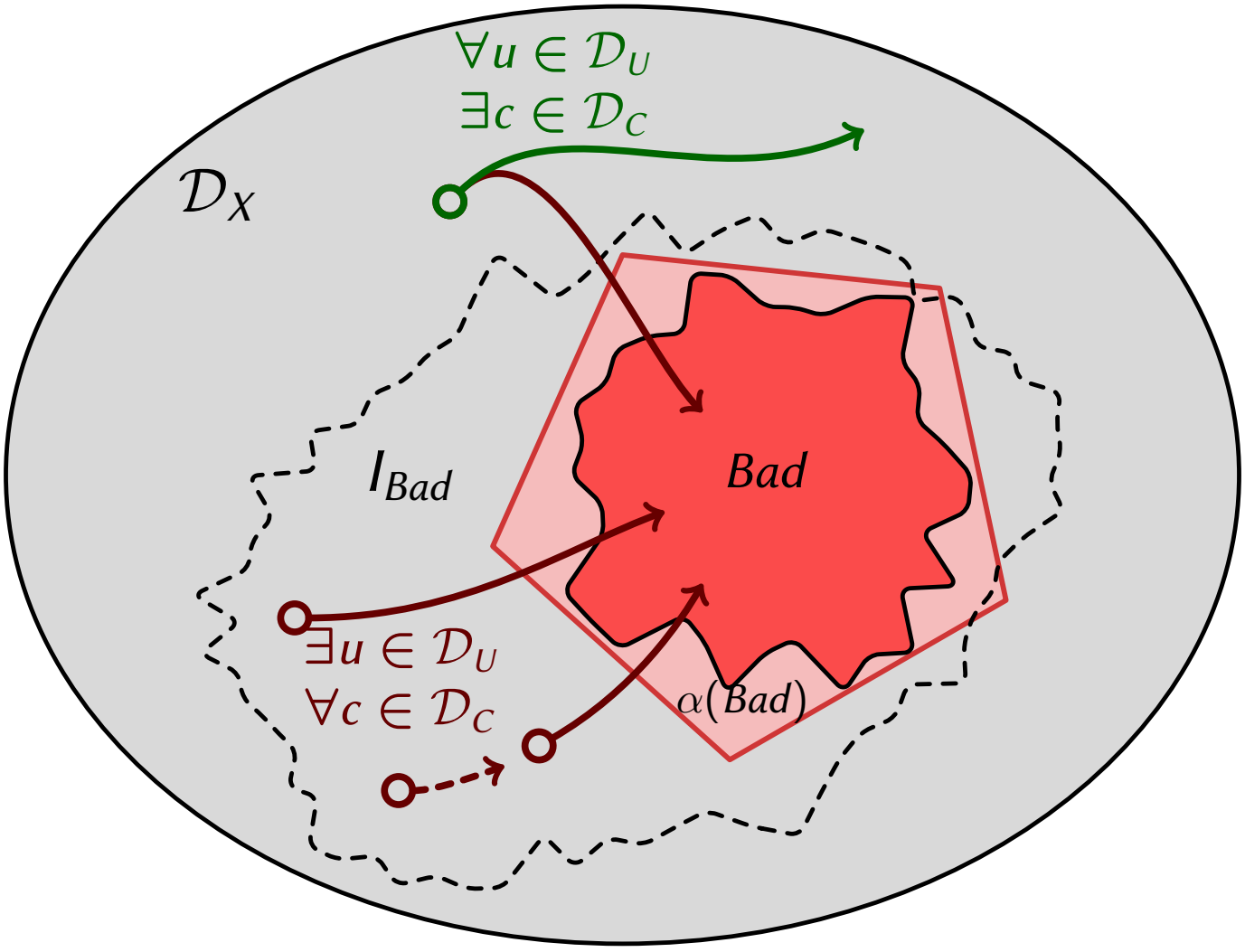
Infinite Case: Computing $I'_{Bad} (\supseteq I_{Bad})$



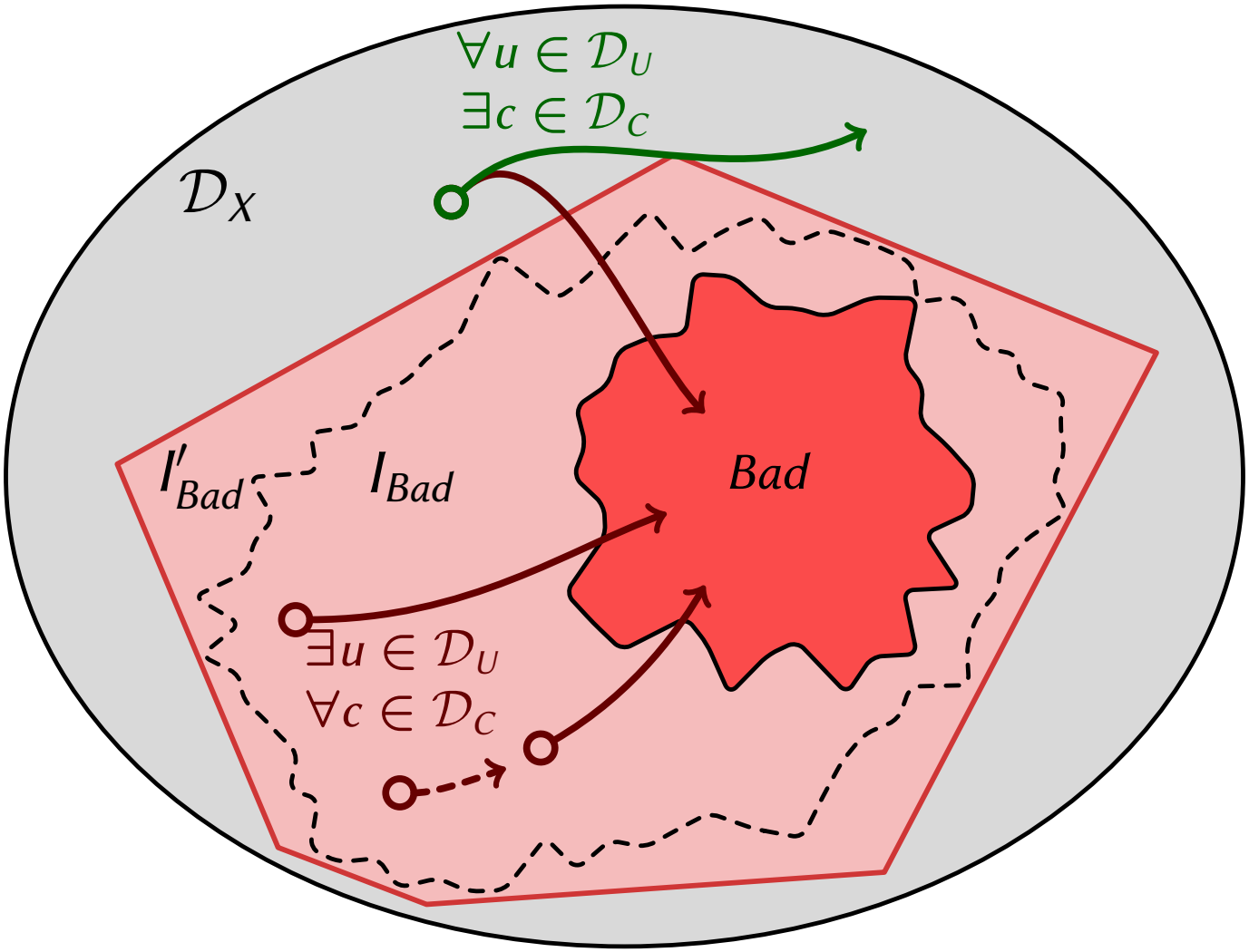
$$I_{Bad} = \text{coreach}_u(Bad) \qquad \text{coreach}_u(B) \stackrel{\text{def}}{=} \text{lfp}\,(\lambda\beta.B \vee \text{pre}_u(\beta))$$

$$\text{pre}_u(B) \stackrel{\text{def}}{=} \exists_U, \forall_C, (T^{-1}(B) \wedge A)$$

Infinite Case: Computing $I'_{Bad} (\supseteq I_{Bad})$



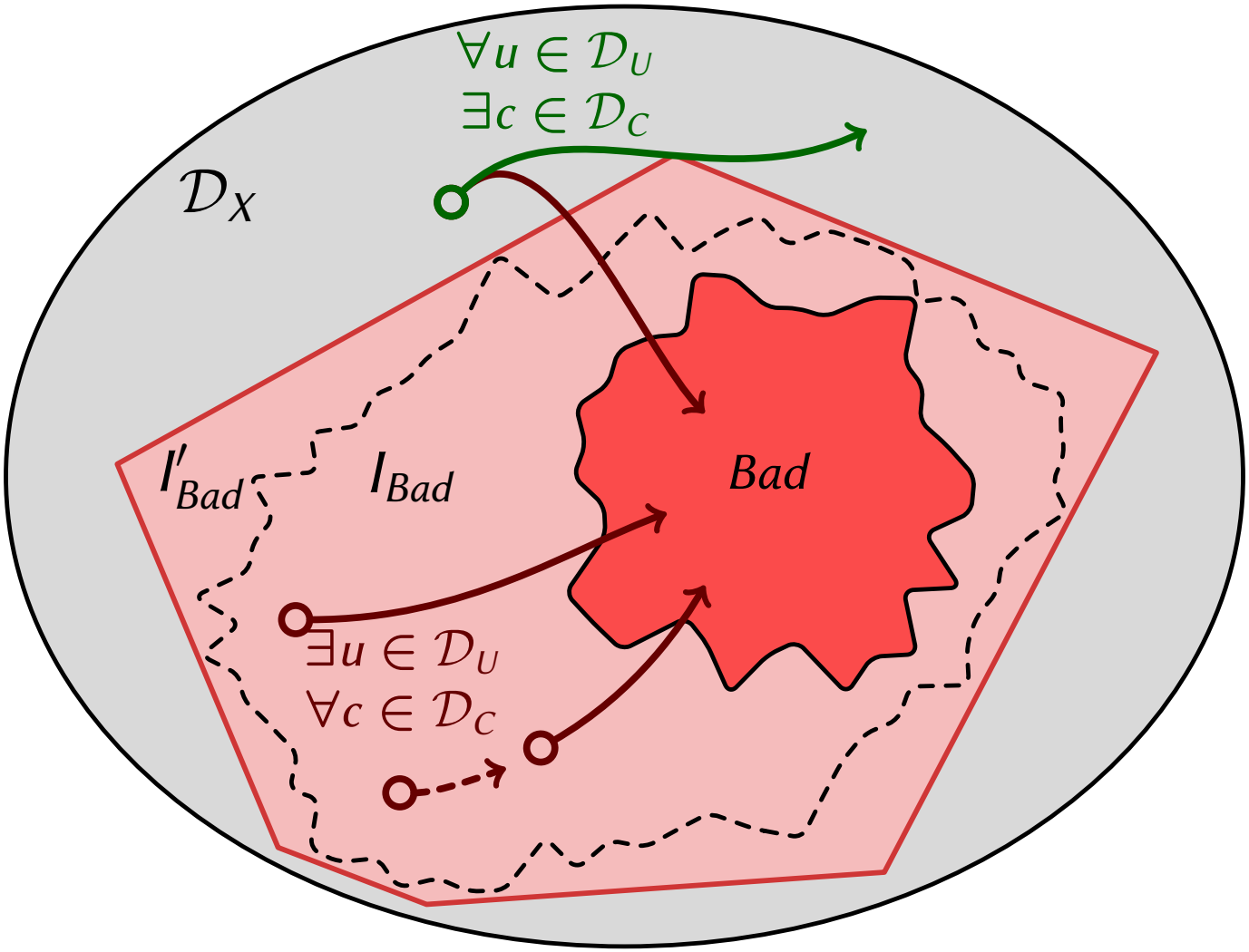
Infinite Case: Computing $I'_{Bad} (\supseteq I_{Bad})$



$$I'_{Bad} = \gamma \circ \text{coreach}^{\sharp}_{u} \circ \alpha(Bad) \qquad \text{coreach}^{\sharp}_{u}(B) \stackrel{\text{def}}{=} \text{lfp}\left(\lambda\beta.B \sqcup \text{pre}^{\sharp}_{u}(\beta)\right)$$

$$\text{pre}^{\sharp}_{u}(B) \stackrel{\text{def}}{=} \exists^{\sharp}_{U}, \forall^{\sharp}_{C}, (T^{\sharp -1}(B) \sqcap \alpha(A))$$

Infinite Case: Computing $I'_{Bad} (\supseteq I_{Bad})$



$$I'_{Bad} = \gamma \circ \text{coreach}_u^{\nabla} \circ \alpha(Bad) \qquad \text{coreach}_u^{\nabla}(B) \stackrel{\text{def}}{=} \text{lfp}\left(\lambda\beta.B \nabla \left(\beta \sqcup \text{pre}_u^{\sharp}(\beta)\right)\right)$$

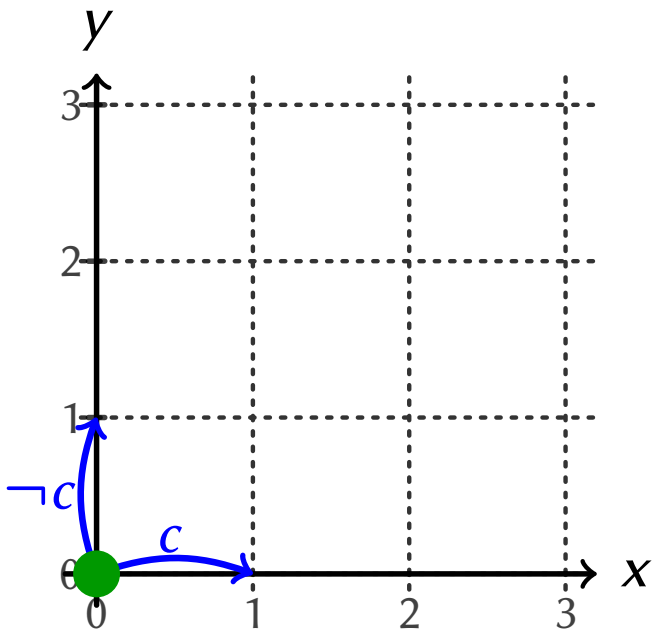
$$\text{pre}_u^{\sharp}(B) \stackrel{\text{def}}{=} \exists_U^{\sharp}, \forall_C^{\sharp}, (T^{\sharp -1}(B) \sqcap \alpha(A))$$

Infinite Case: Example Execution

Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } \quad c \quad \text{else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \text{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$



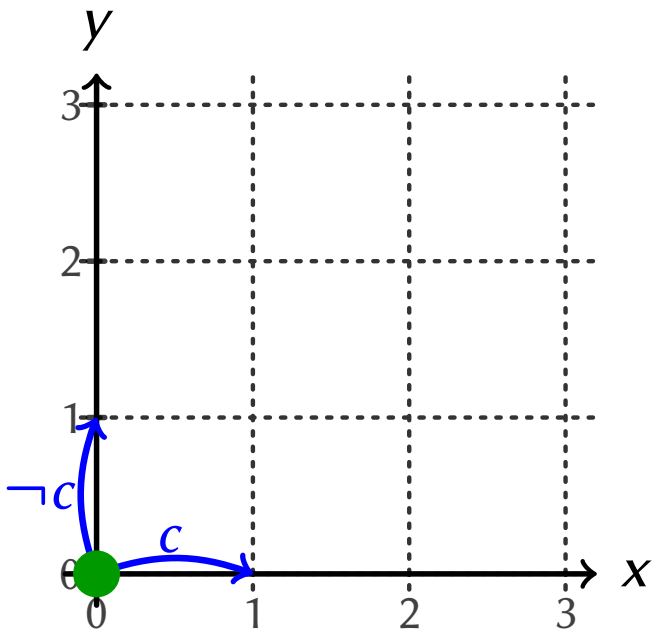
Infinite Case: Example Execution

Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } c \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \text{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = v_{x<2}$ $\rightsquigarrow Bad = (\neg v_{x<2})$



Infinite Case: Example Execution

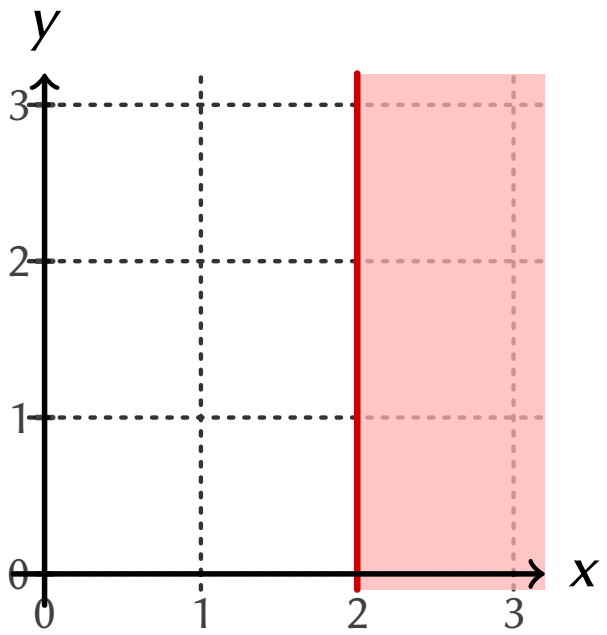
Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } c \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \mathrm{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = v_{x<2}$ $\rightsquigarrow Bad = (\neg v_{x<2})$

- $\ell_0 = \alpha(\neg v_{x<2}) = x \geqslant 2$



Infinite Case: Example Execution

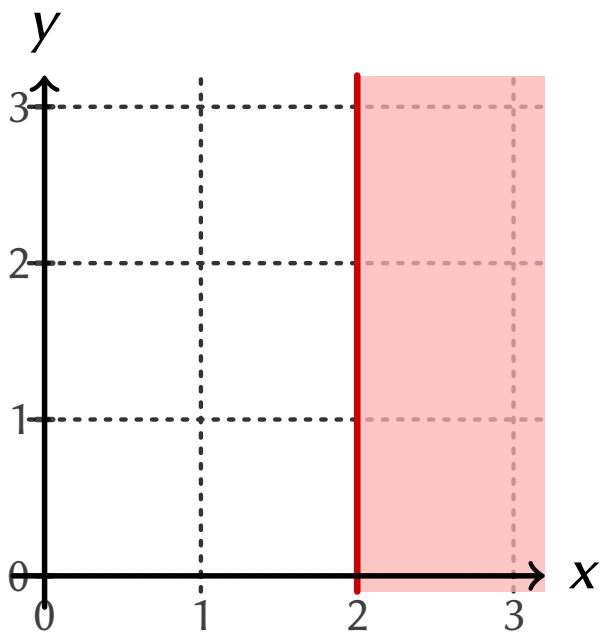
Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } \ c \ \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \text{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = v_{x<2}$ $\rightsquigarrow Bad = (\neg v_{x<2})$

- $\ell_0 = \alpha(\neg v_{x<2}) = x \geqslant 2$
- $\ell_\infty = \text{coreach}_u^{\nabla}(\ell_0) =$
  $\text{lfp}\left(\lambda\beta.\ell_0 \nabla (\beta \sqcup \text{pre}_u^\sharp(\beta))\right)$
  - $\ell_0 \nabla \left(\ell_0 \sqcup \text{pre}_u^\sharp(\ell_0)\right)$



Infinite Case: Example Execution

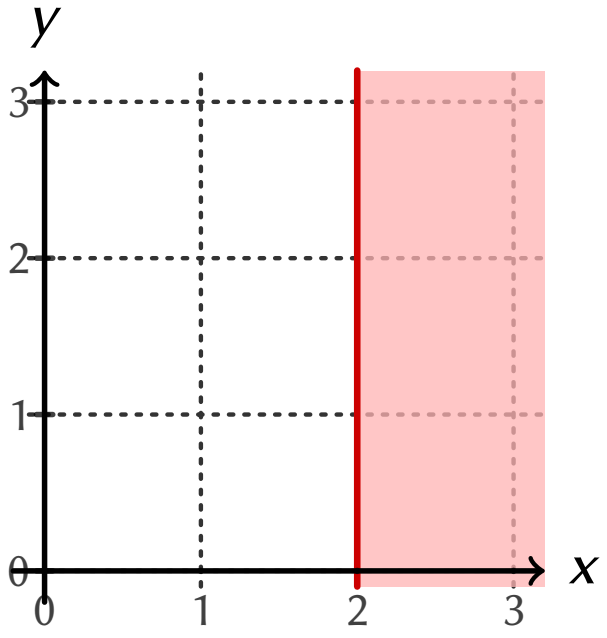
Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } \ c \ \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \mathrm{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = v_{x<2}$ $\rightsquigarrow Bad = (\neg v_{x<2})$

- $\ell_0 = \alpha(\neg v_{x<2}) = x \geqslant 2$
- $\ell_\infty = \mathrm{coreach}_u^{\nabla}(\ell_0) =$
  - $\ell_0 \nabla (\ell_0 \sqcup \mathrm{pre}_u^\sharp(\ell_0))$
  - $\mathrm{pre}_u^\sharp(\ell_0) = \forall_c, \exists_\varnothing, (T^{\sharp -1}(\ell_0) \sqcap \alpha(\mathrm{tt}))$



Infinite Case: Example Execution

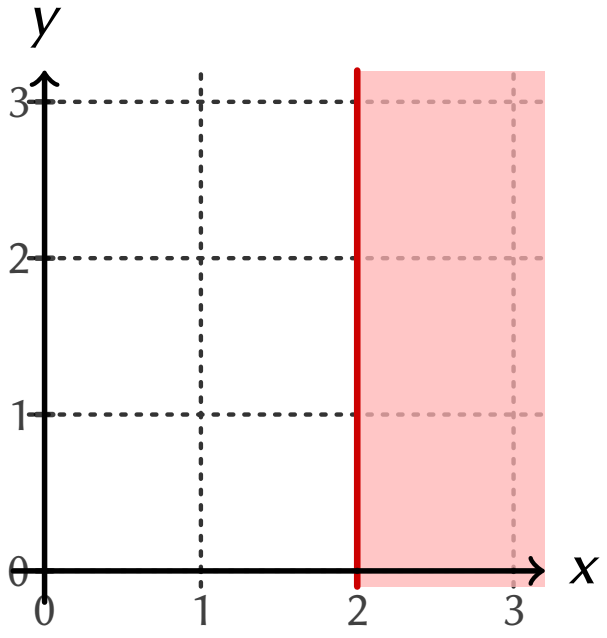
Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } \ c \ \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \mathrm{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = v_{x<2}$ $\rightsquigarrow Bad = (\neg v_{x<2})$

- $\ell_0 = \alpha(\neg v_{x<2}) = x \geqslant 2$
- $\ell_\infty = \mathrm{coreach}_u^{\nabla}(\ell_0) =$
  - $\ell_0 \nabla (\ell_0 \sqcup \mathrm{pre}_u^\sharp(\ell_0))$
  - $\mathrm{pre}_u^\sharp(\ell_0) = \forall_c, \exists_\varnothing, (T^{\sharp -1}(\ell_0) \sqcap \alpha(\mathrm{tt}))$



Infinite Case: Example Execution

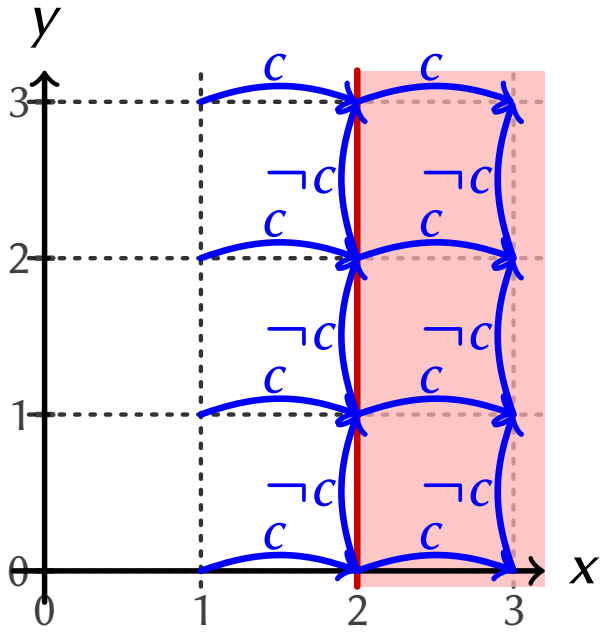
Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } \ c \ \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \text{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = v_{x<2}$ $\rightsquigarrow Bad = (\neg v_{x<2})$

- $\ell_0 = \alpha(\neg v_{x<2}) = x \geqslant 2$
- $\ell_\infty = \text{coreach}_u^{\nabla}(\ell_0) =$
  - $\ell_0 \nabla (\ell_0 \sqcup \text{pre}_u^\sharp(\ell_0))$
  - $\text{pre}_u^\sharp(\ell_0) = \forall_c, \exists_\varnothing, (T^{\sharp -1}(\ell_0) \sqcap \alpha(\text{tt}))$
    $= \forall_c, \exists_\varnothing, (c \wedge x \geqslant 1) \vee (\neg c \wedge x \geqslant 2)$



Infinite Case: Example Execution

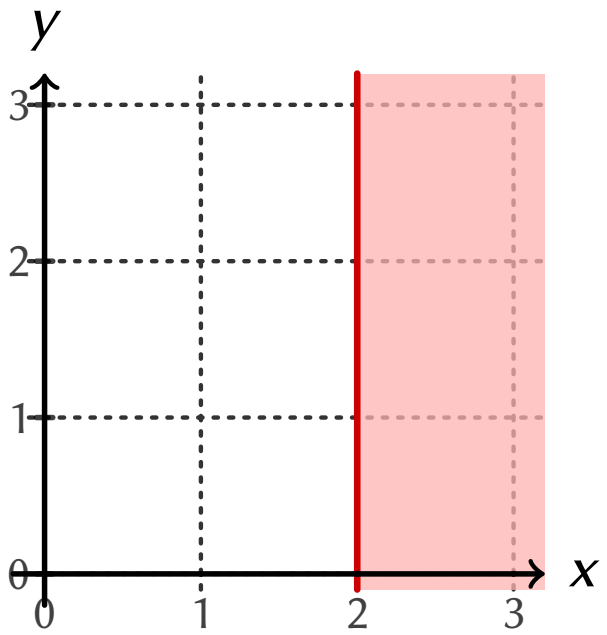
Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } \ c \ \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \mathrm{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = v_{x<2}$ $\rightsquigarrow Bad = (\neg v_{x<2})$

- $\ell_0 = \alpha(\neg v_{x<2}) = x \geqslant 2$
- $\ell_\infty = \mathrm{coreach}_u^{\nabla}(\ell_0) =$
  - $\ell_0 \nabla \left(\ell_0 \sqcup \mathrm{pre}_u^\sharp(\ell_0)\right) = x \geqslant 2 = \ell_0$
  - $\mathrm{pre}_u^\sharp(\ell_0) = \forall_c, \exists_\varnothing, (T^{\sharp -1}(\ell_0) \sqcap \alpha(\mathrm{tt}))$
    $= \forall_c, \exists_\varnothing, (c \wedge x \geqslant 1) \vee (\neg c \wedge x \geqslant 2)$
    $= x \geqslant 2$



Infinite Case: Example Execution

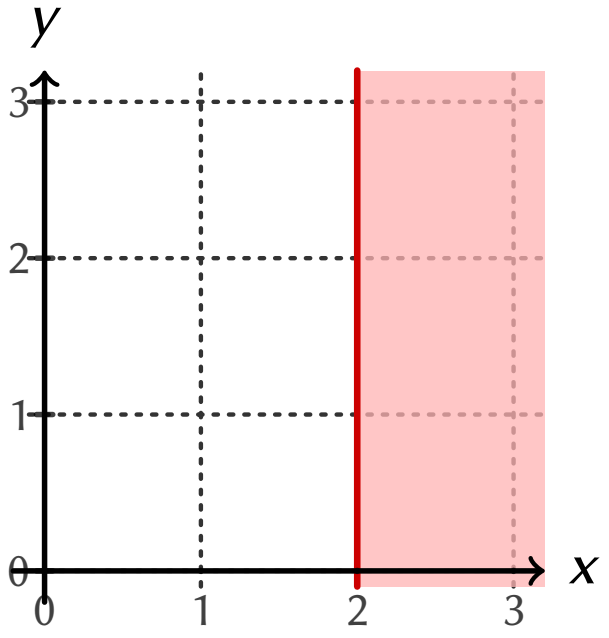
Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } \ c \ \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \text{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = v_{x<2}$ $\rightsquigarrow Bad = (\neg v_{x<2})$

- $\ell_0 = \alpha(\neg v_{x<2}) = x \geqslant 2$
- $\ell_\infty = \text{coreach}_u^{\nabla}(\ell_0) = x \geqslant 2$
  - $\ell_0 \nabla (\ell_0 \sqcup \text{pre}_u^\sharp(\ell_0)) = x \geqslant 2 = \ell_0$
  - $\text{pre}_u^\sharp(\ell_0) = \forall_c, \exists_\varnothing, (T^{\sharp -1}(\ell_0) \sqcap \alpha(\text{tt}))$
    $= \forall_c, \exists_\varnothing, (c \wedge x \geqslant 1) \vee (\neg c \wedge x \geqslant 2)$
    $= x \geqslant 2$



Infinite Case: Example Execution

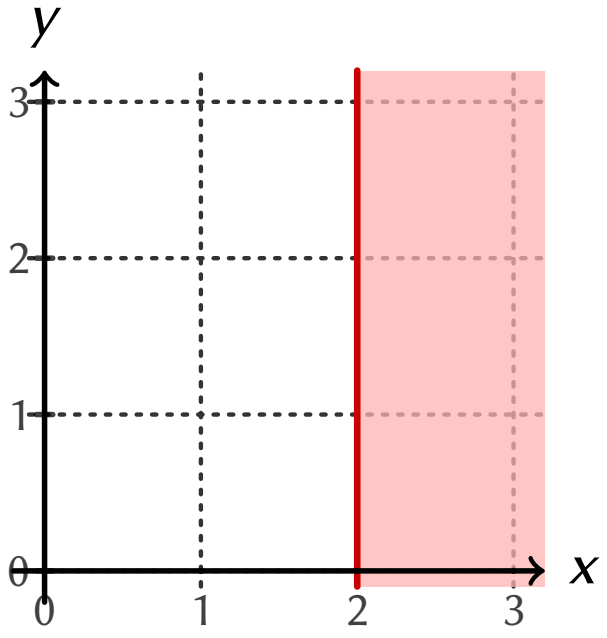
Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } \ c \ \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \text{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = v_{x<2}$ $\rightsquigarrow Bad = (\neg v_{x<2})$

- $\ell_0 = \alpha(\neg v_{x<2}) = x \geqslant 2$
- $\ell_\infty = \text{coreach}_u^{\nabla}(\ell_0) = x \geqslant 2$
  - $\ell_0 \nabla (\ell_0 \sqcup \text{pre}_u^\sharp(\ell_0)) = x \geqslant 2 = \ell_0$
  - $\text{pre}_u^\sharp(\ell_0) = \forall_c, \exists_\varnothing, (T^{\sharp -1}(\ell_0) \sqcap \alpha(\text{tt}))$
    $= \forall_c, \exists_\varnothing, (c \wedge x \geqslant 1) \vee (\neg c \wedge x \geqslant 2)$
    $= x \geqslant 2$
- $\alpha(\Theta_0) \not\sqsubseteq \ell_\infty \rightsquigarrow$ Success
- $I'_{Bad} = \gamma(\ell_\infty) = \neg v_{x<2}$



Infinite Case: Example Execution

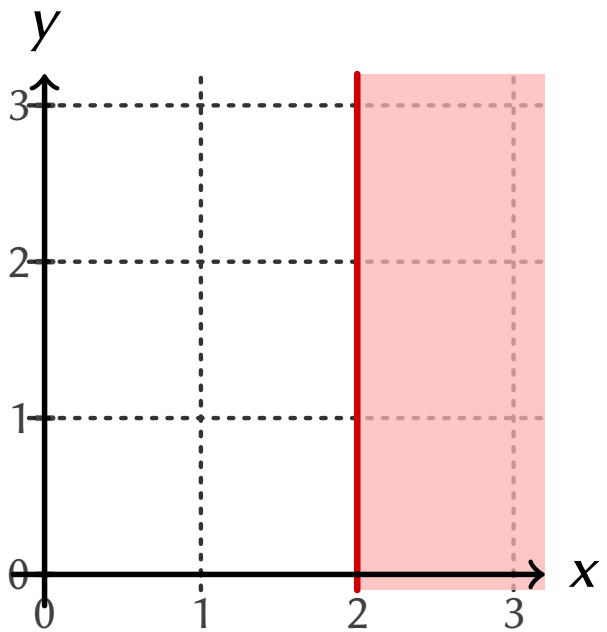
Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } c \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \mathrm{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = v_{x<2}$ $\rightsquigarrow Bad = (\neg v_{x<2})$

- $\ell_0 = \alpha(\neg v_{x<2}) = x \geqslant 2$
- $\ell_\infty = \mathrm{coreach}_u^{\nabla}(\ell_0) = x \geqslant 2$
  - $\ell_0 \nabla (\ell_0 \sqcup \mathrm{pre}_u^\sharp(\ell_0)) = x \geqslant 2 = \ell_0$
  - $\mathrm{pre}_u^\sharp(\ell_0) = \forall_c, \exists_\varnothing, (T^{\sharp -1}(\ell_0) \sqcap \alpha(\mathrm{tt}))$
    $= \forall_c, \exists_\varnothing, (c \wedge x \geqslant 1) \vee (\neg c \wedge x \geqslant 2)$
    $= x \geqslant 2$
- $\alpha(\Theta_0) \not\sqsubseteq \ell_\infty \rightsquigarrow$ Success
- $I'_{Bad} = \gamma(\ell_\infty) = \neg v_{x<2}$
- $K_\Phi = T^{-1}(v_{x<2}) \wedge \mathrm{tt}$



Infinite Case: Example Execution

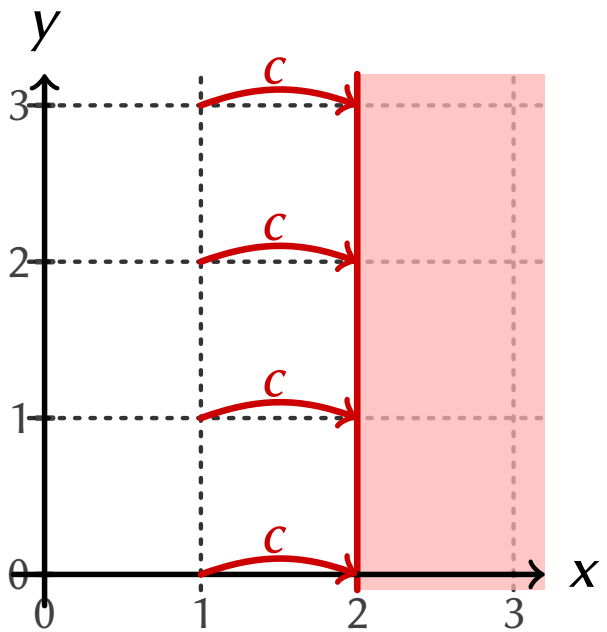
Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } c \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \mathrm{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = v_{x<2}$ $\rightsquigarrow Bad = (\neg v_{x<2})$

- $\ell_0 = \alpha(\neg v_{x<2}) = x \geqslant 2$
- $\ell_\infty = \mathrm{coreach}_u^{\nabla}(\ell_0) = x \geqslant 2$
  - $\ell_0 \nabla (\ell_0 \sqcup \mathrm{pre}_u^\sharp(\ell_0)) = x \geqslant 2 = \ell_0$
  - $\mathrm{pre}_u^\sharp(\ell_0) = \forall_c, \exists_\varnothing, (T^{\sharp -1}(\ell_0) \sqcap \alpha(\mathrm{tt}))$
    $= \forall_c, \exists_\varnothing, (c \wedge x \geqslant 1) \vee (\neg c \wedge x \geqslant 2)$
    $= x \geqslant 2$
- $\alpha(\Theta_0) \not\sqsubseteq \ell_\infty \rightsquigarrow$ Success
- $I'_{Bad} = \gamma(\ell_\infty) = \neg v_{x<2}$
- $K_\Phi = (\neg c \wedge v_{x \leqslant 1}) \vee (c \wedge v_{x \leqslant 0})$



Outline

Main Limitations of the Previous Solution

Universal Quantification on Controllable Variables

- $\mathcal{D}_C = \mathbb{B}^m \quad \rightsquigarrow \quad$ BDD Operation

Main Limitations of the Previous Solution

Universal Quantification on Controllable Variables

- $\mathcal{D}_C = \mathbb{B}^m \quad \rightsquigarrow \quad$ BDD Operation

Assumed Convexity of *Bad*

e.g., $\Phi\left(\langle x \rangle\right) = \left(v_{x \geqslant 0} \wedge v_{x \leqslant 10}\right) \quad \rightsquigarrow \quad Bad = \neg\Phi = \left(\neg v_{x \geqslant 0} \vee \neg v_{x \leqslant 10}\right)$

- Yet $\alpha\left(\neg v_{x \geqslant 0} \vee \neg v_{x \leqslant 10}\right) = \top$ with Usual Numerical Abstract Domains
- Naive, Unsound Solution for Non-convex *Bad*
  - Split *Bad* into a Disjunction of Convex Clauses Bad_i
  - Compute the Controller using $I'_{Bad} = \bigvee_i \left(\gamma \circ \text{coreach}_u^{\nabla} \circ \alpha(Bad_i)\right)$
- Yet: Reachable States May Become Non-reactive ("Deadlocks")!

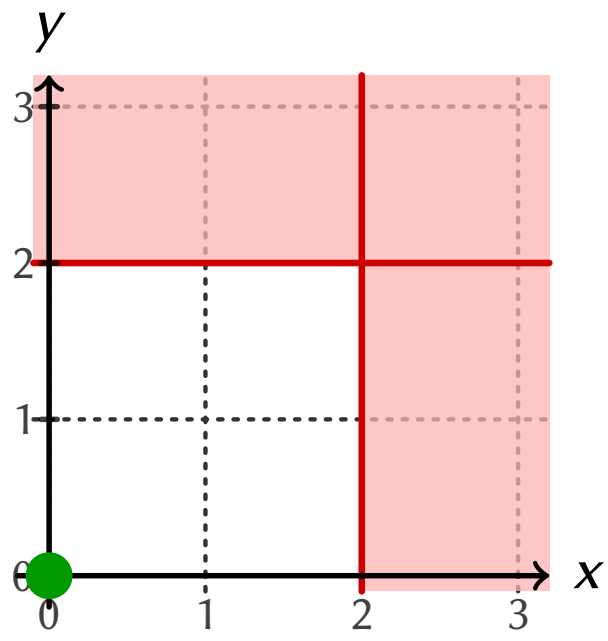
Example Deadlocking Case

Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } c \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \mathrm{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = (v_{x<2} \wedge v_{y<2})$ $\rightsquigarrow Bad = (\neg v_{x<2} \vee \neg v_{y<2})$



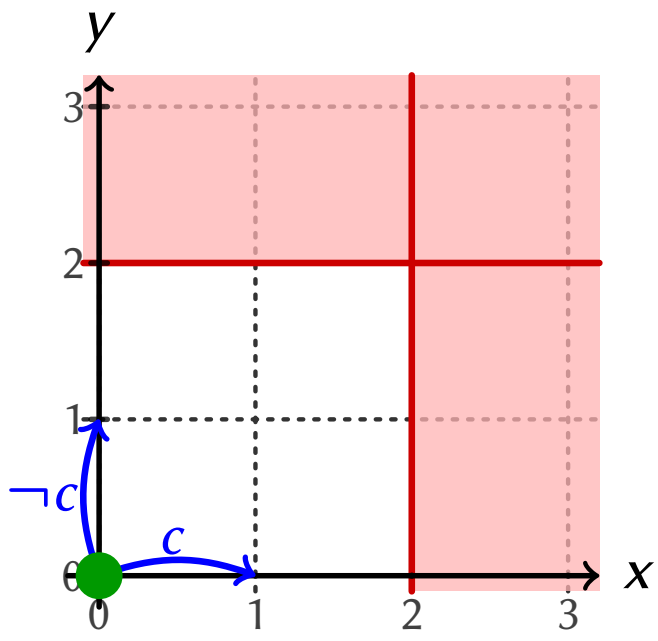
Example Deadlocking Case

Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } c \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \text{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = (v_{x<2} \wedge v_{y<2})$ $\rightsquigarrow Bad = (\neg v_{x<2} \vee \neg v_{y<2})$



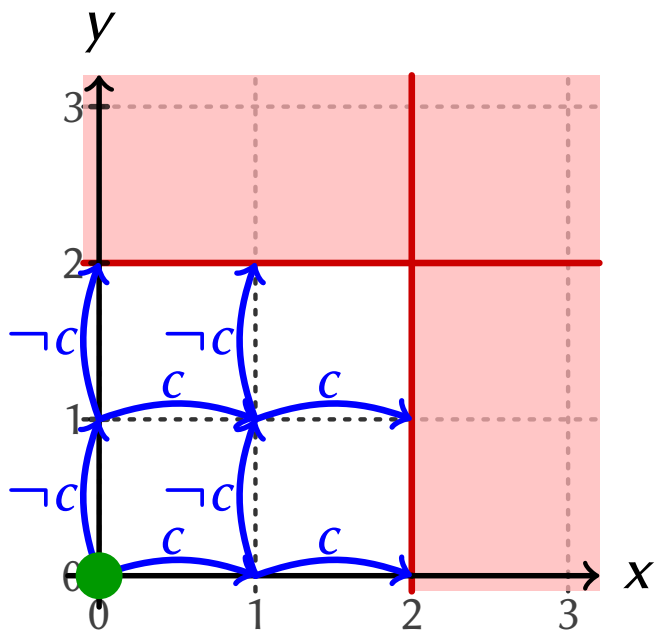
Example Deadlocking Case

Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } c \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \mathrm{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = (v_{x<2} \wedge v_{y<2})$ $\rightsquigarrow Bad = (\neg v_{x<2} \vee \neg v_{y<2})$



Example Deadlocking Case

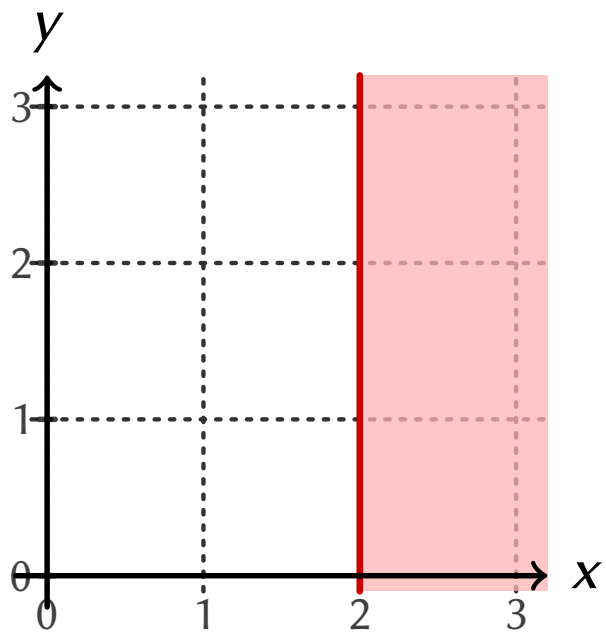
Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } c \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \mathrm{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = (v_{x<2} \wedge v_{y<2})$ $\rightsquigarrow Bad = (\neg v_{x<2} \vee \neg v_{y<2})$

- $\gamma \circ \mathrm{coreach}_u^{\nabla} \circ \alpha(\neg v_{x<2}) = ?$



Example Deadlocking Case

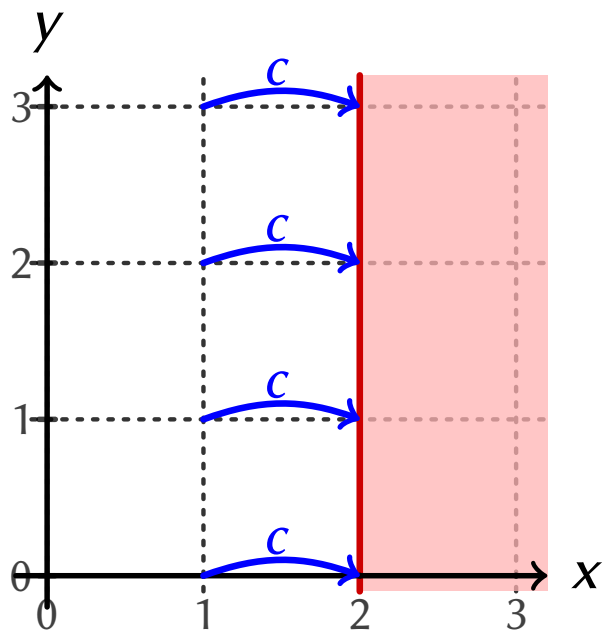
Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } c \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \mathrm{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = (v_{x<2} \wedge v_{y<2})$ $\rightsquigarrow Bad = (\neg v_{x<2} \vee \neg v_{y<2})$

- $\gamma \circ \mathrm{coreach}_u^{\nabla} \circ \alpha(\neg v_{x<2}) = ?$



Example Deadlocking Case

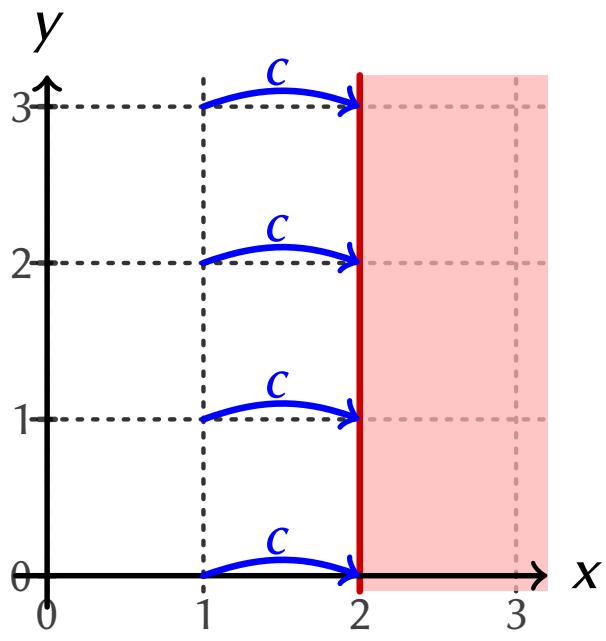
Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } c \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \mathrm{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = (v_{x<2} \wedge v_{y<2})$ $\rightsquigarrow Bad = (\neg v_{x<2} \vee \neg v_{y<2})$

- $\gamma \circ \mathrm{coreach}_u^{\nabla} \circ \alpha(\neg v_{x<2}) = x \geqslant 2$



Example Deadlocking Case

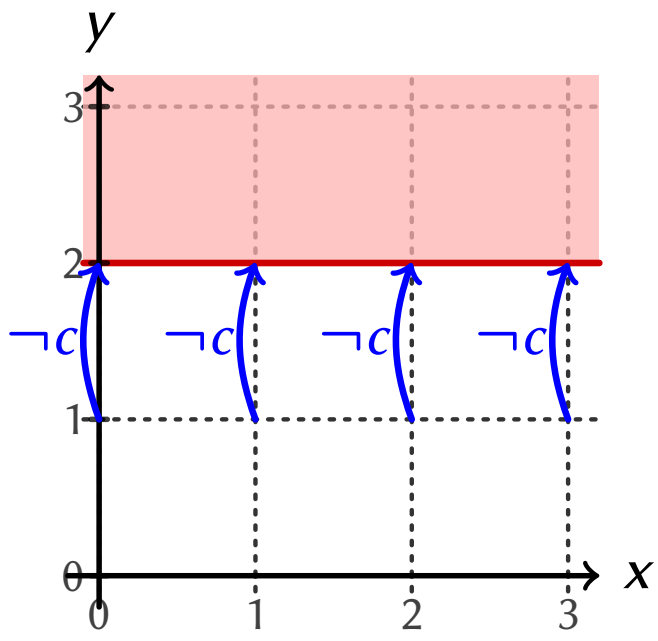
Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } c \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \text{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = (v_{x<2} \wedge v_{y<2})$ $\rightsquigarrow Bad = (\neg v_{x<2} \vee \neg v_{y<2})$

- $\gamma \circ \text{coreach}_u^{\nabla} \circ \alpha(\neg v_{x<2}) = x \geqslant 2$
- $\gamma \circ \text{coreach}_u^{\nabla} \circ \alpha(\neg v_{y<2}) = y \geqslant 2$



Example Deadlocking Case

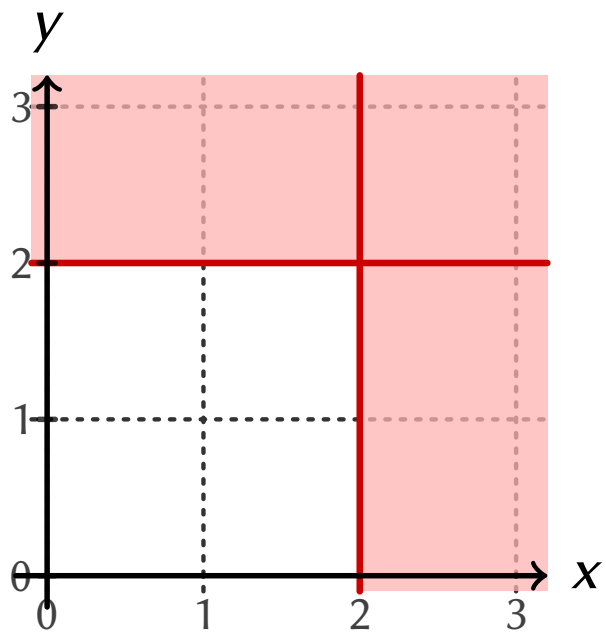
Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } c \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \mathrm{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = (v_{x<2} \wedge v_{y<2})$ $\rightsquigarrow Bad = (\neg v_{x<2} \vee \neg v_{y<2})$

- $\gamma \circ \mathrm{coreach}_u^{\nabla} \circ \alpha(\neg v_{x<2}) = x \geqslant 2$
- $\gamma \circ \mathrm{coreach}_u^{\nabla} \circ \alpha(\neg v_{y<2}) = y \geqslant 2$
- $I'_{Bad} = (\neg v_{x<2} \vee \neg v_{y<2})$



Example Deadlocking Case

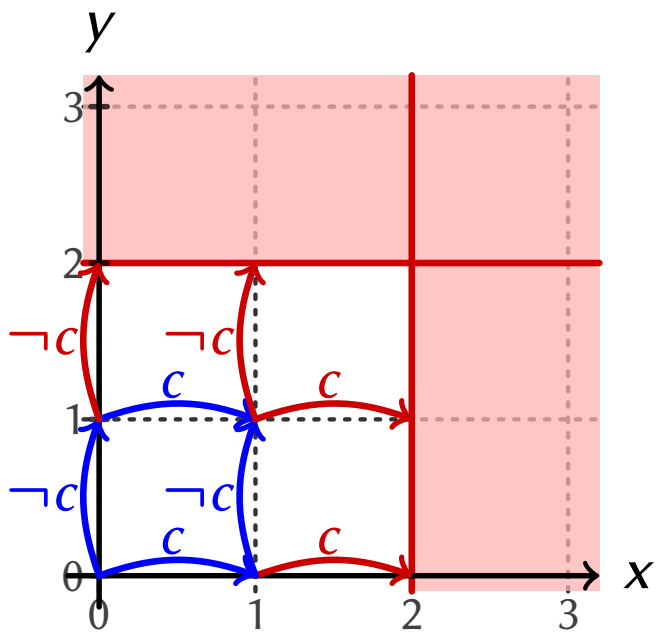
Example Program

$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } \ c \ \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \mathrm{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = (v_{x<2} \wedge v_{y<2})$ $\rightsquigarrow Bad = (\neg v_{x<2} \vee \neg v_{y<2})$

- $\gamma \circ \mathrm{coreach}_u^{\nabla} \circ \alpha(\neg v_{x<2}) = x \geqslant 2$
- $\gamma \circ \mathrm{coreach}_u^{\nabla} \circ \alpha(\neg v_{y<2}) = y \geqslant 2$
- $I'_{Bad} = (\neg v_{x<2} \vee \neg v_{y<2})$
- $K_\Phi = T^{-1}(v_{x<2} \wedge v_{y<2}) \wedge \mathrm{tt}$
  $= (\neg c \wedge v_{x\leqslant 1} \wedge v_{y\leqslant 0}) \vee$
  $= (\ \ c \wedge v_{x\leqslant 0} \wedge v_{y\leqslant 1})$



Example Deadlocking Case

Example Program

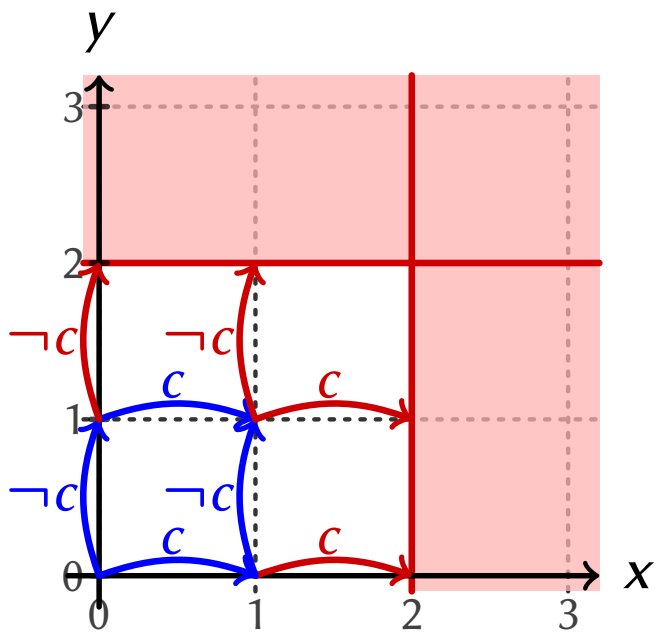
$$X = \langle x, y \rangle, \mathcal{D}_X = \mathbb{Z}^2 \qquad C = \langle c \rangle, \mathcal{D}_C = \mathbb{B} \qquad U = \varnothing$$

$$T = \begin{cases} x' = x + 1 \text{ if } c \text{ else } x \\ y' = y + 1 \text{ if } \neg c \text{ else } y \end{cases} \qquad \begin{aligned} A(\langle x, y, c \rangle) &= \mathrm{tt} \\ \Theta_0(\langle x, y \rangle) &= (x = 0 \wedge y = 0) \end{aligned}$$

Invariant: $\Phi(\langle x, y \rangle) = (v_{x<2} \wedge v_{y<2})$ $\rightsquigarrow Bad = (\neg v_{x<2} \vee \neg v_{y<2})$

- $\gamma \circ \mathrm{coreach}_u^{\nabla} \circ \alpha(\neg v_{x<2}) = x \geqslant 2$
- $\gamma \circ \mathrm{coreach}_u^{\nabla} \circ \alpha(\neg v_{y<2}) = y \geqslant 2$
- $I'_{Bad} = (\neg v_{x<2} \vee \neg v_{y<2})$
- $K_\Phi = T^{-1}(v_{x<2} \wedge v_{y<2}) \wedge \mathrm{tt}$
  $= (\neg c \wedge v_{x\leqslant 1} \wedge v_{y\leqslant 0}) \vee$
  $= (\quad c \wedge v_{x\leqslant 0} \wedge v_{y\leqslant 1})$

$\rightsquigarrow$ Deadlock in $X = \langle 1, 1 \rangle$!



Handling Disjunctions & Avoiding Deadlocks: Basic Idea

Powerset Domain Extension

- Handling of Disjunctions for Non-convex *Bad*
- Enforcing Reactivity of the Result

Proposals

1. Previous Algorithm with Powerset Extension
2. Custom, Double Fixpoint Algorithm

Disjunctive Abstract Domain

Finite Powerset Domain over Λ (Bagnara *et al*[2])

$\langle \wp(\Lambda), \Subset, \bigoplus, \varnothing \rangle$, $\alpha_\wp$ and $\gamma_\wp$ such that: $\wp(\mathcal{D}_X) \underset{\alpha_\wp}{\overset{\gamma_\wp}{\leftrightarrows}} \wp(\Lambda)$

- Non-redundancy ← Forget Non-maximal Elements
- $\Subset$ ← Hoare Partial Order
- $\alpha_\wp$ ← Decomposition into Convex Disjuncts, & Abstraction
- $\bigoplus$ ← Set Union Ensuring Non-redundancy

[2] Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. "Widening Operators for Powerset Domains". In: *Int. J. Softw. Tools Technol. Transf.* 8.4 (Aug. 2006), pp. 449–466.

Disjunctive Abstract Domain

Finite Powerset Domain over Λ (Bagnara *et al*[2])

$\langle \wp(\Lambda), \Subset, \oplus, \varnothing \rangle$, $\alpha_\wp$ and $\gamma_\wp$ such that: $\wp(\mathcal{D}_X) \underset{\alpha_\wp}{\overset{\gamma_\wp}{\leftrightarrows}} \wp(\Lambda)$

- Non-redundancy ← Forget Non-maximal Elements
- $\Subset$ ← Hoare Partial Order
- $\alpha_\wp$ ← Decomposition into Convex Disjuncts, & Abstraction
- $\oplus$ ← Set Union Ensuring Non-redundancy
- Universal Quantification Performed in the Concrete Domain

$$\alpha_\wp\left(\forall_C, \gamma_\wp\left(\ell\right)\right)$$

[2] Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. "Widening Operators for Powerset Domains". In: *Int. J. Softw. Tools Technol. Transf.* 8.4 (Aug. 2006), pp. 449–466.

Disjunctive Abstract Domain

Finite Powerset Domain over Λ (Bagnara *et al*[2])

$\langle \wp(\Lambda), \Subset, \bigoplus, \varnothing \rangle$, $\alpha_\wp$ and $\gamma_\wp$ such that: $\wp(\mathcal{D}_X) \underset{\alpha_\wp}{\overset{\gamma_\wp}{\leftrightarrows}} \wp(\Lambda)$

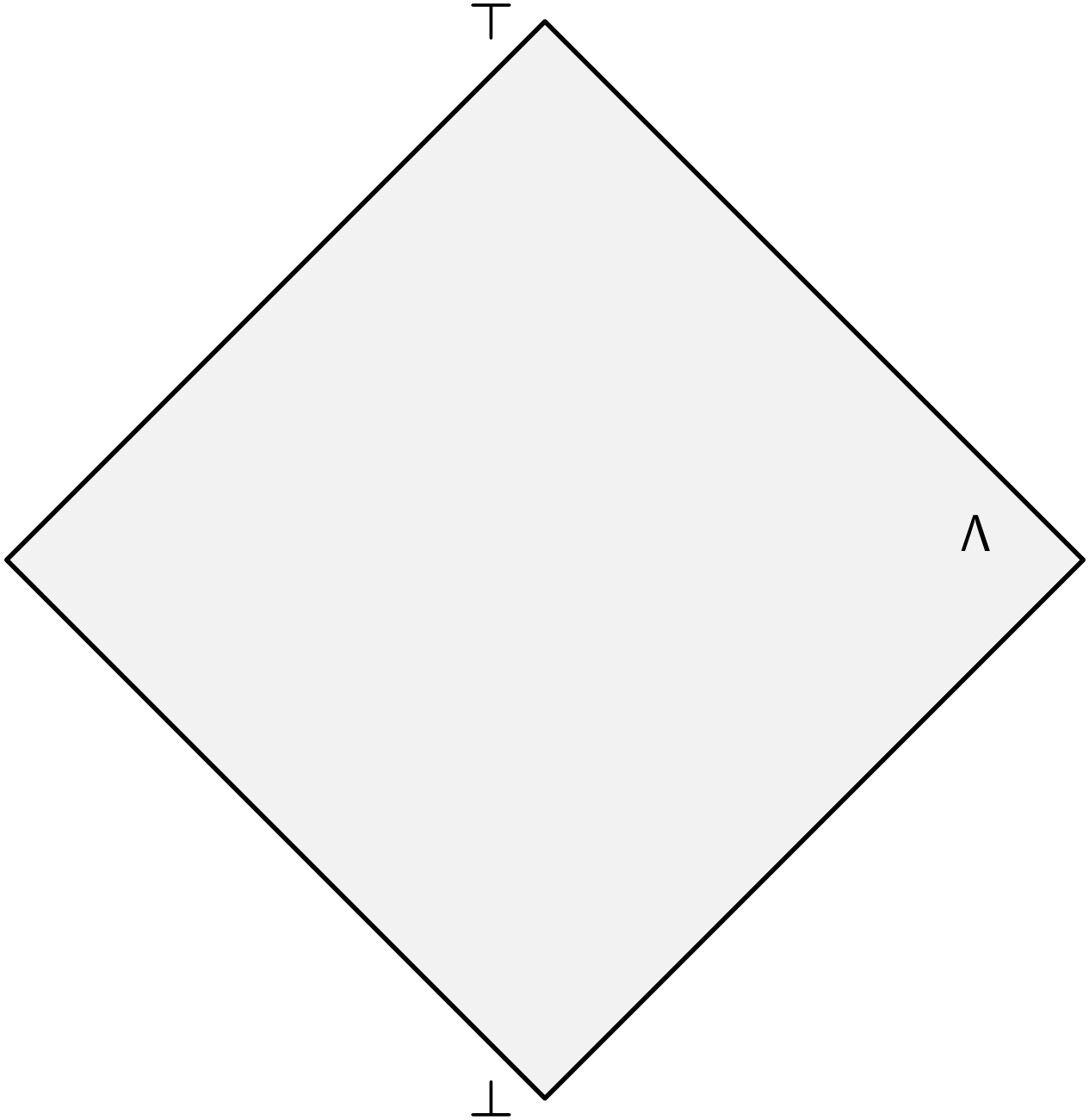
- Non-redundancy ← Forget Non-maximal Elements
- $\Subset$ ← Hoare Partial Order
- $\alpha_\wp$ ← Decomposition into Convex Disjuncts, & Abstraction
- $\bigoplus$ ← Set Union Ensuring Non-redundancy
- Universal Quantification Performed in the Concrete Domain

$$\alpha_\wp\left(\forall_C, \gamma_\wp\left(\ell\right)\right)$$

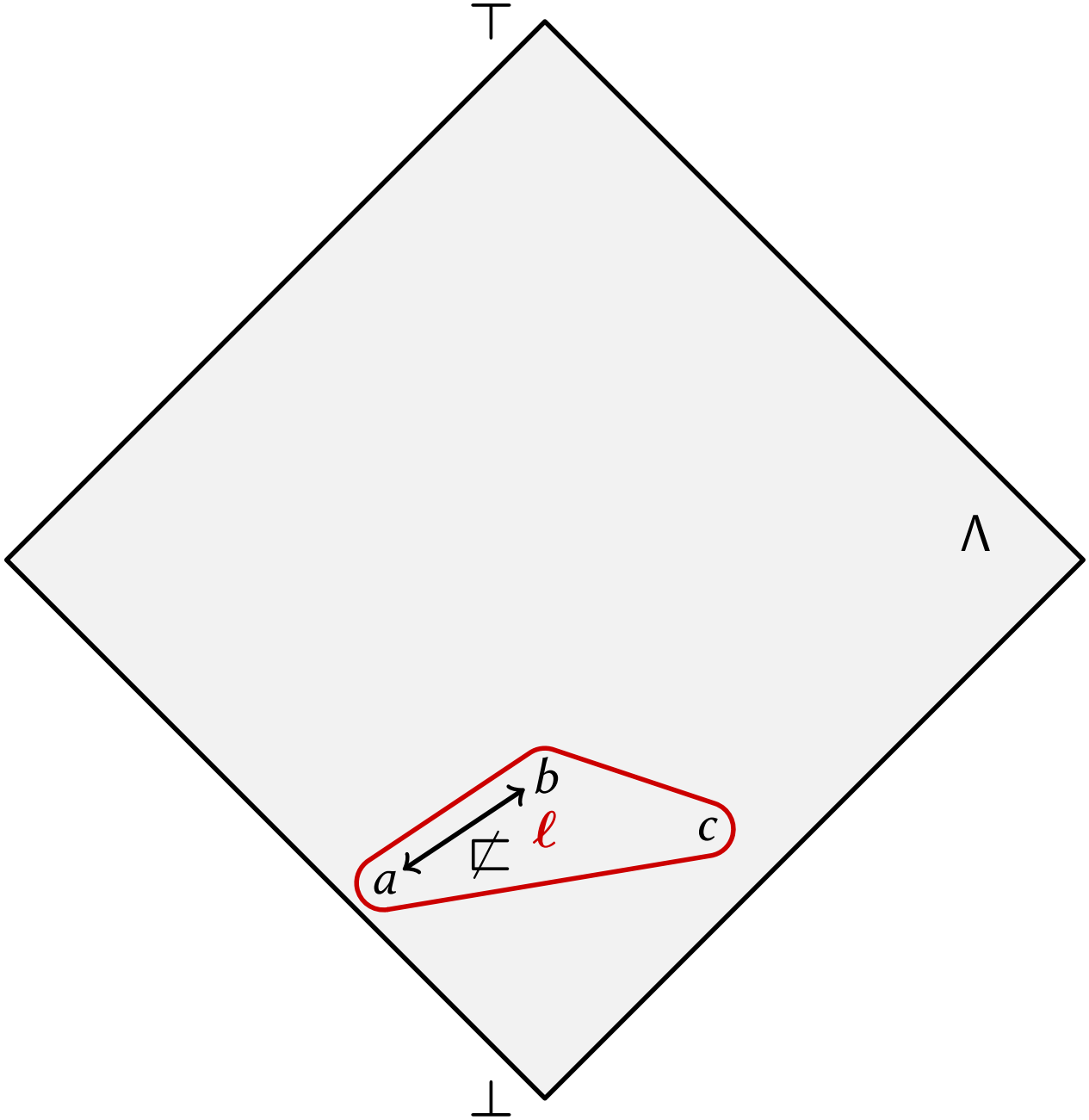
- $\nabla_\wp$ ← Ensuring Convergence
  - Lifting Base Domain Widening
  - Egli-Milner Ordering Enforcement

[2] Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. "Widening Operators for Powerset Domains". In: *Int. J. Softw. Tools Technol. Transf.* 8.4 (Aug. 2006), pp. 449–466.

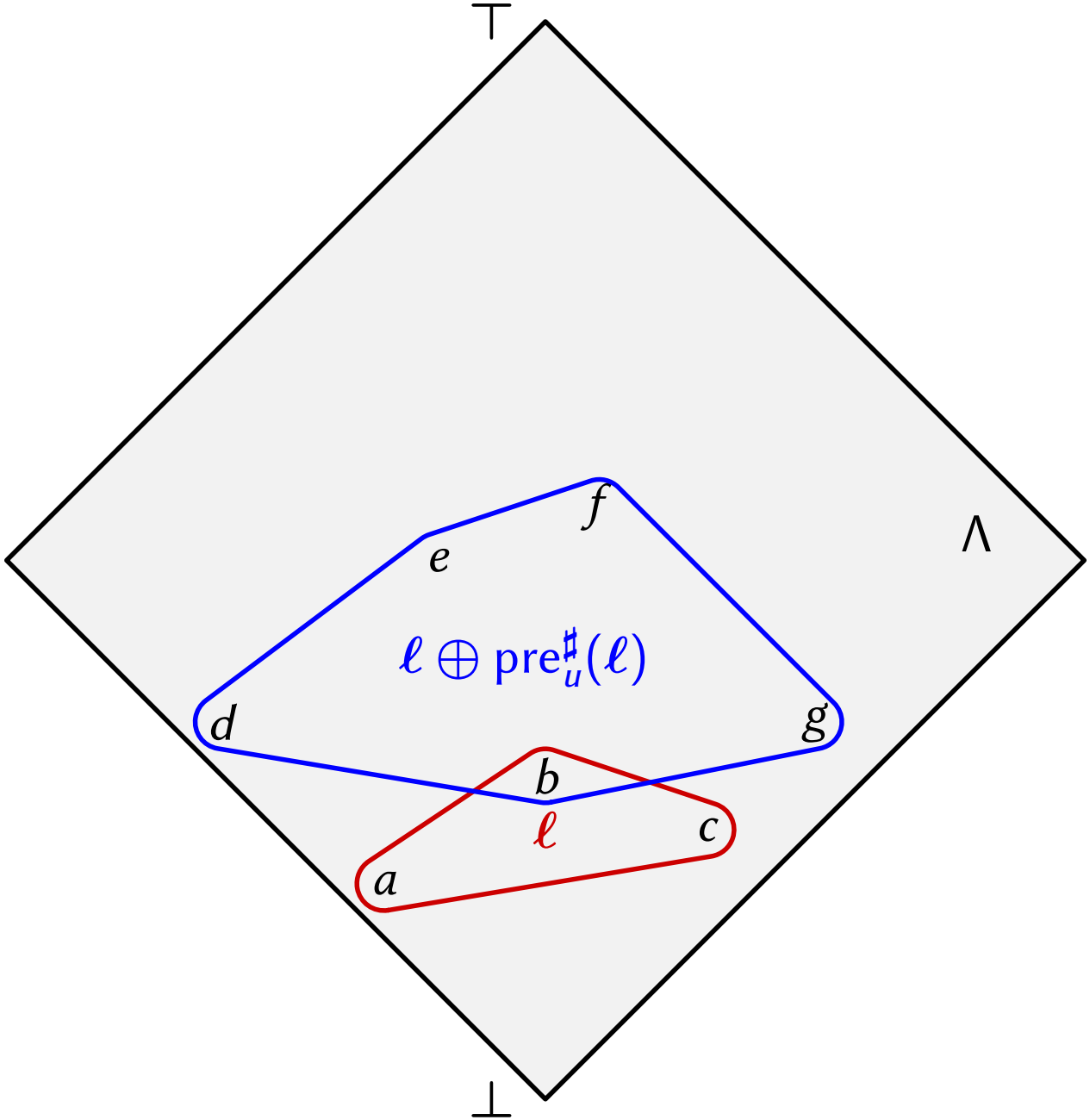
Egli-Milner Ordering Enforcement for Widening in $\wp(\Lambda)$



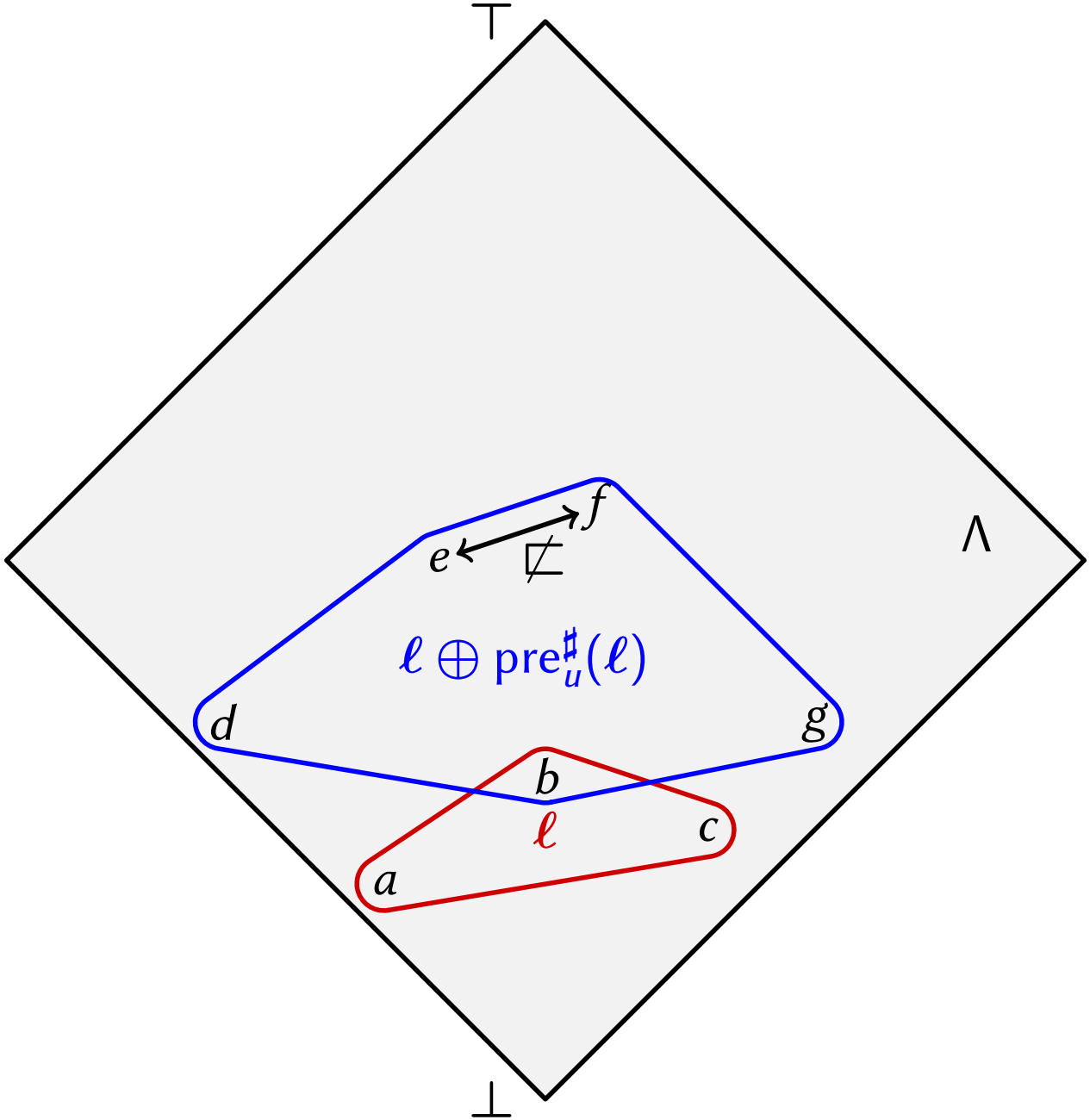
Egli-Milner Ordering Enforcement for Widening in $\wp(\Lambda)$



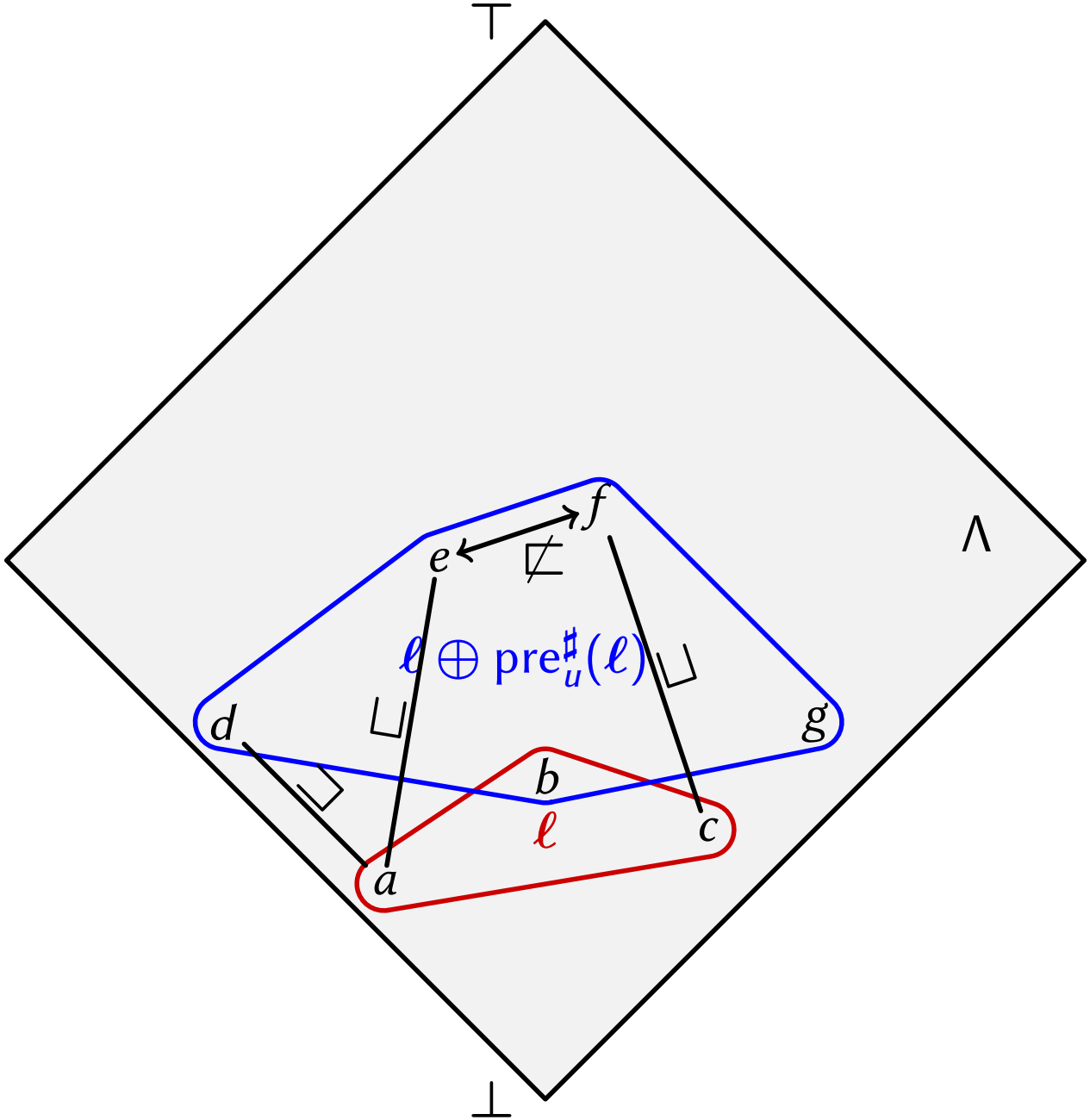
Egli-Milner Ordering Enforcement for Widening in $\wp(\Lambda)$



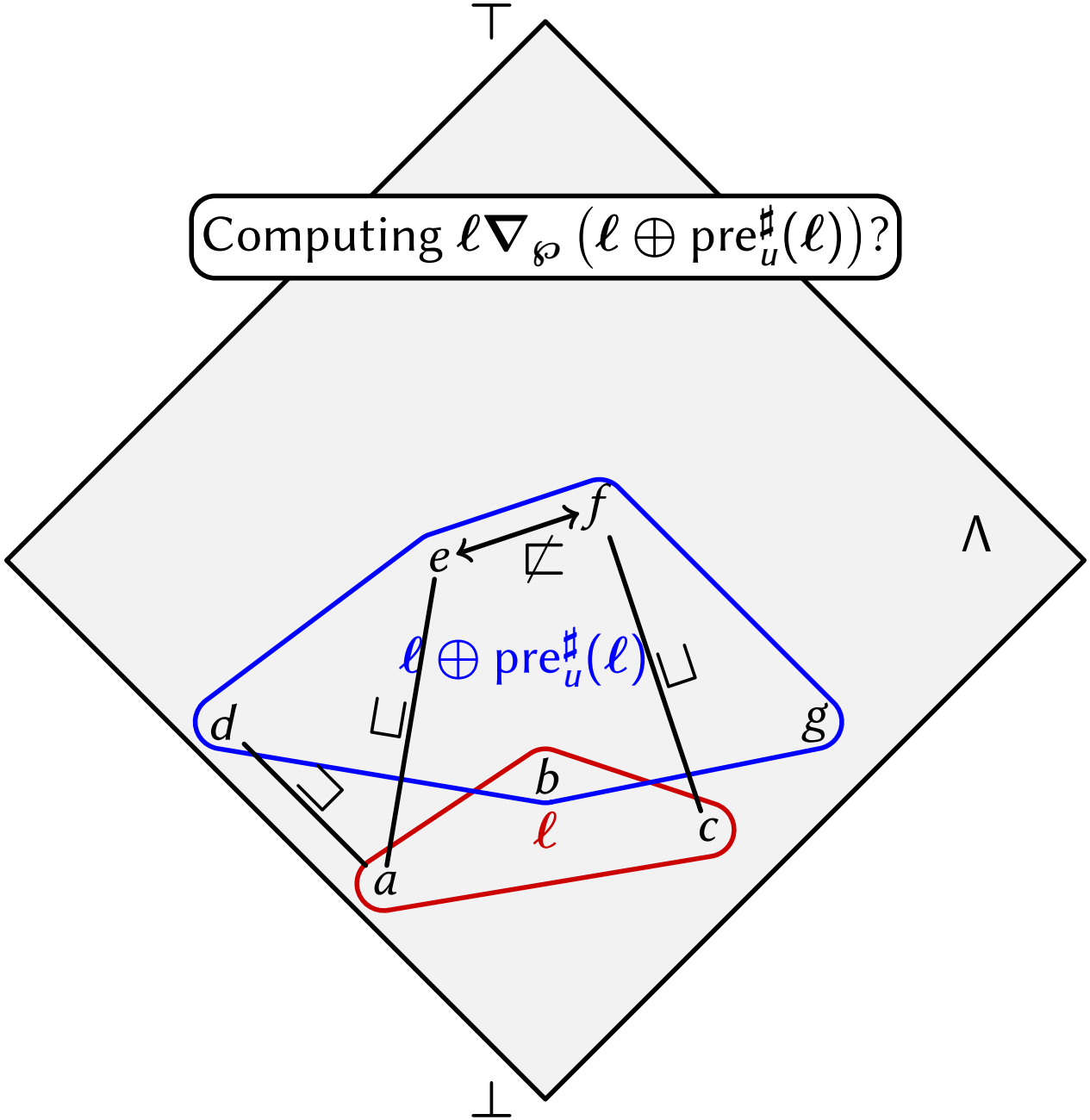
Egli-Milner Ordering Enforcement for Widening in $\wp(\Lambda)$



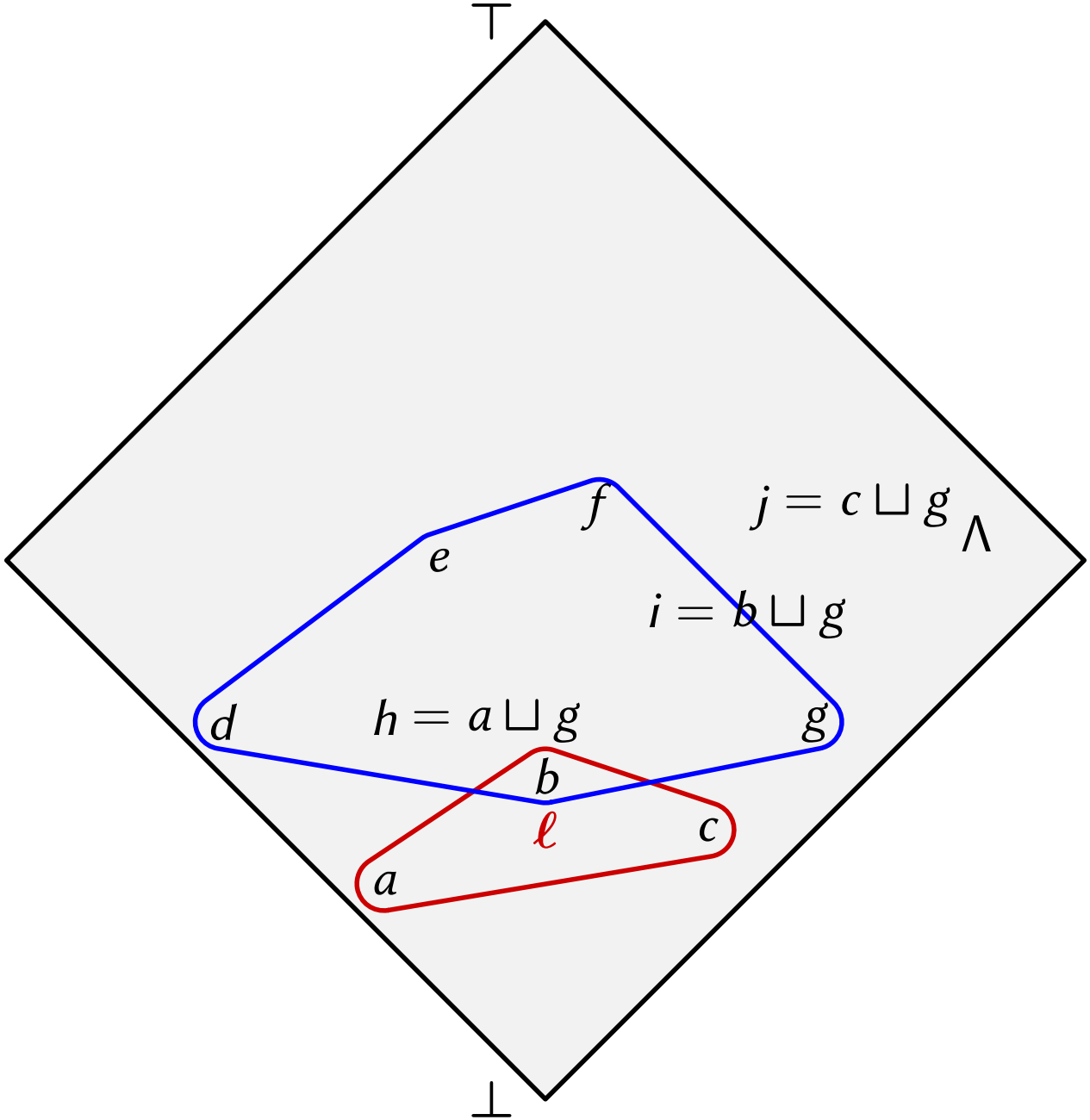
Egli-Milner Ordering Enforcement for Widening in $\wp(\Lambda)$



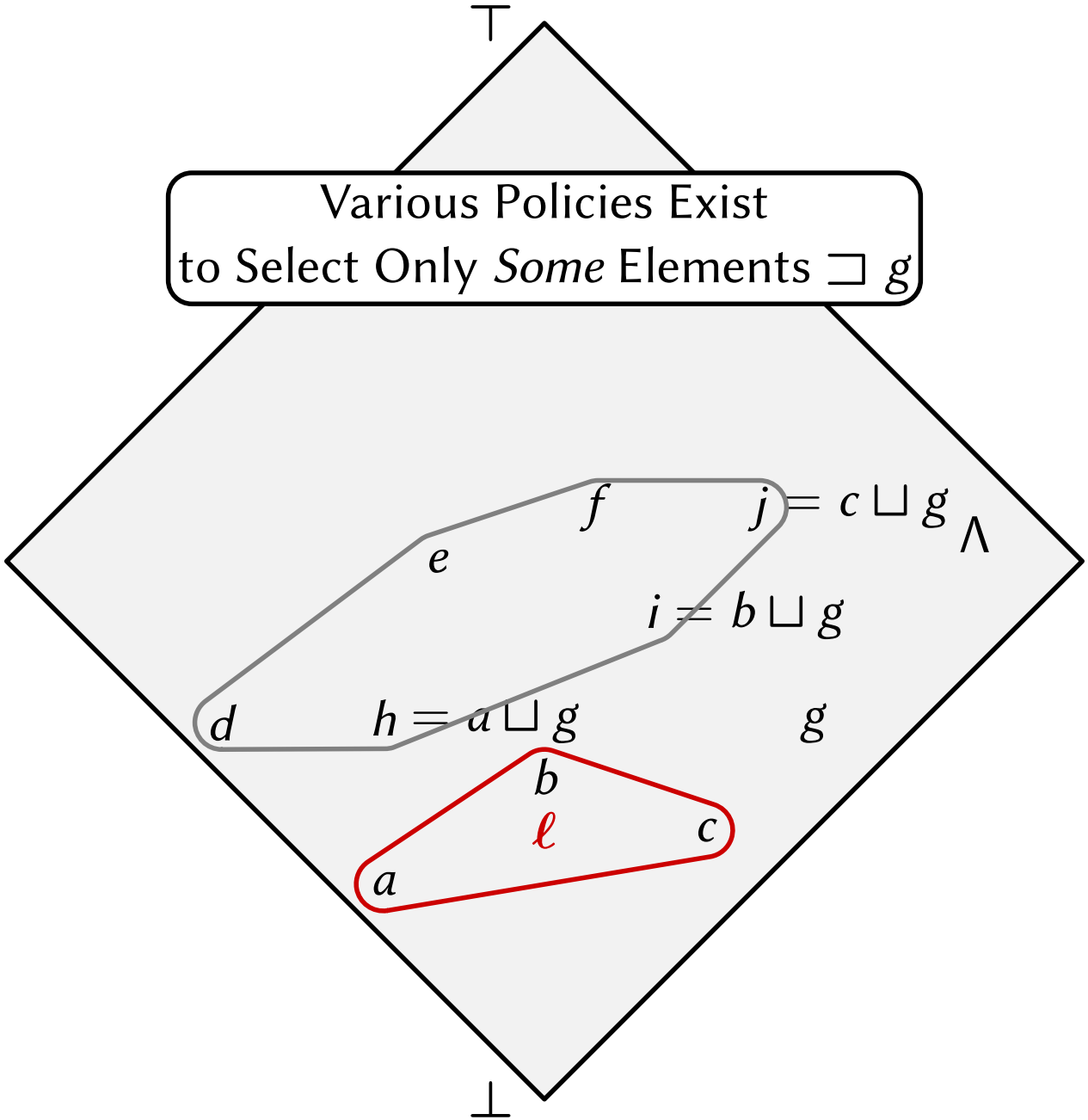
Egli-Milner Ordering Enforcement for Widening in $\wp(\Lambda)$



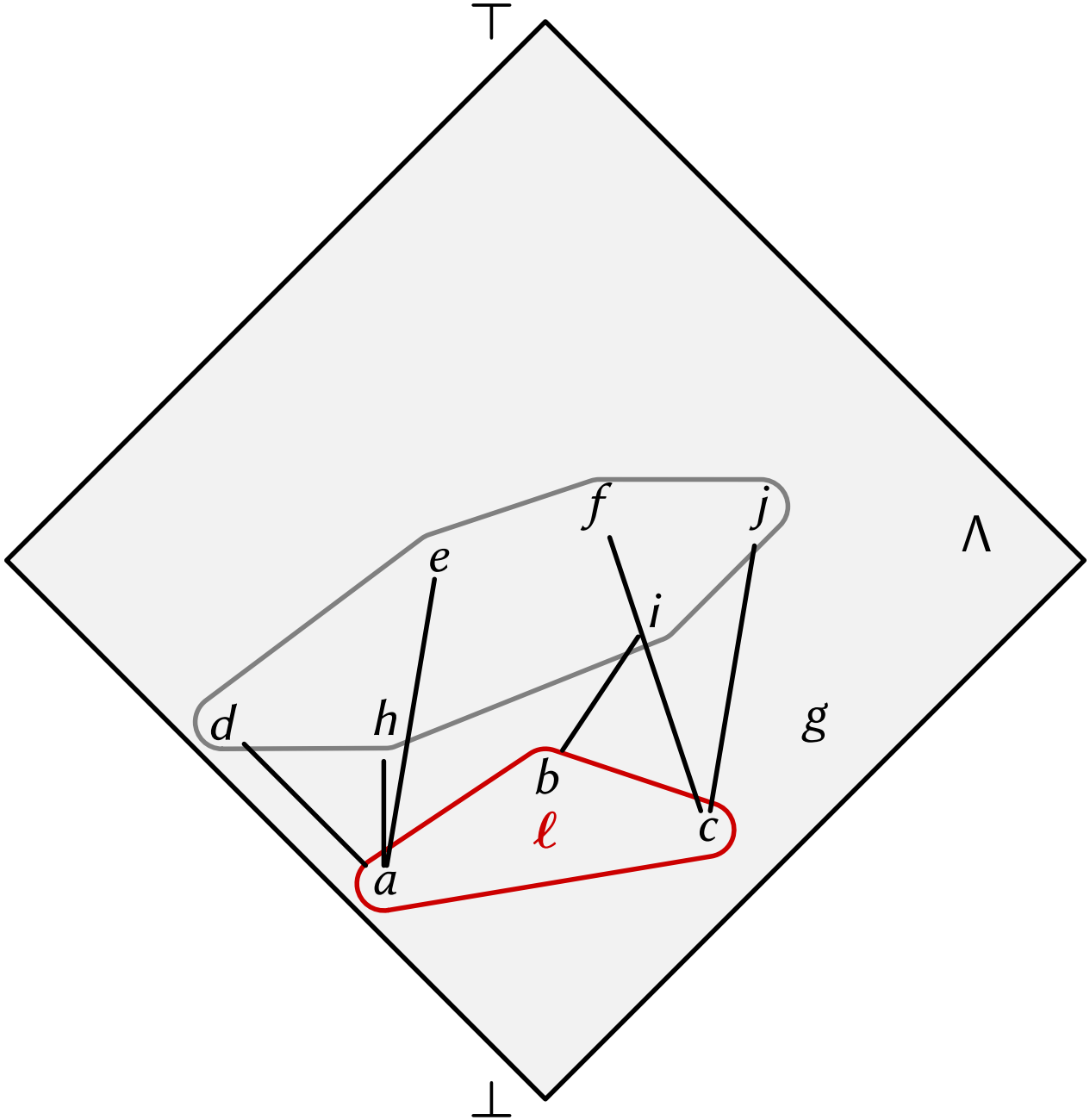
Egli-Milner Ordering Enforcement for Widening in $\wp(\Lambda)$



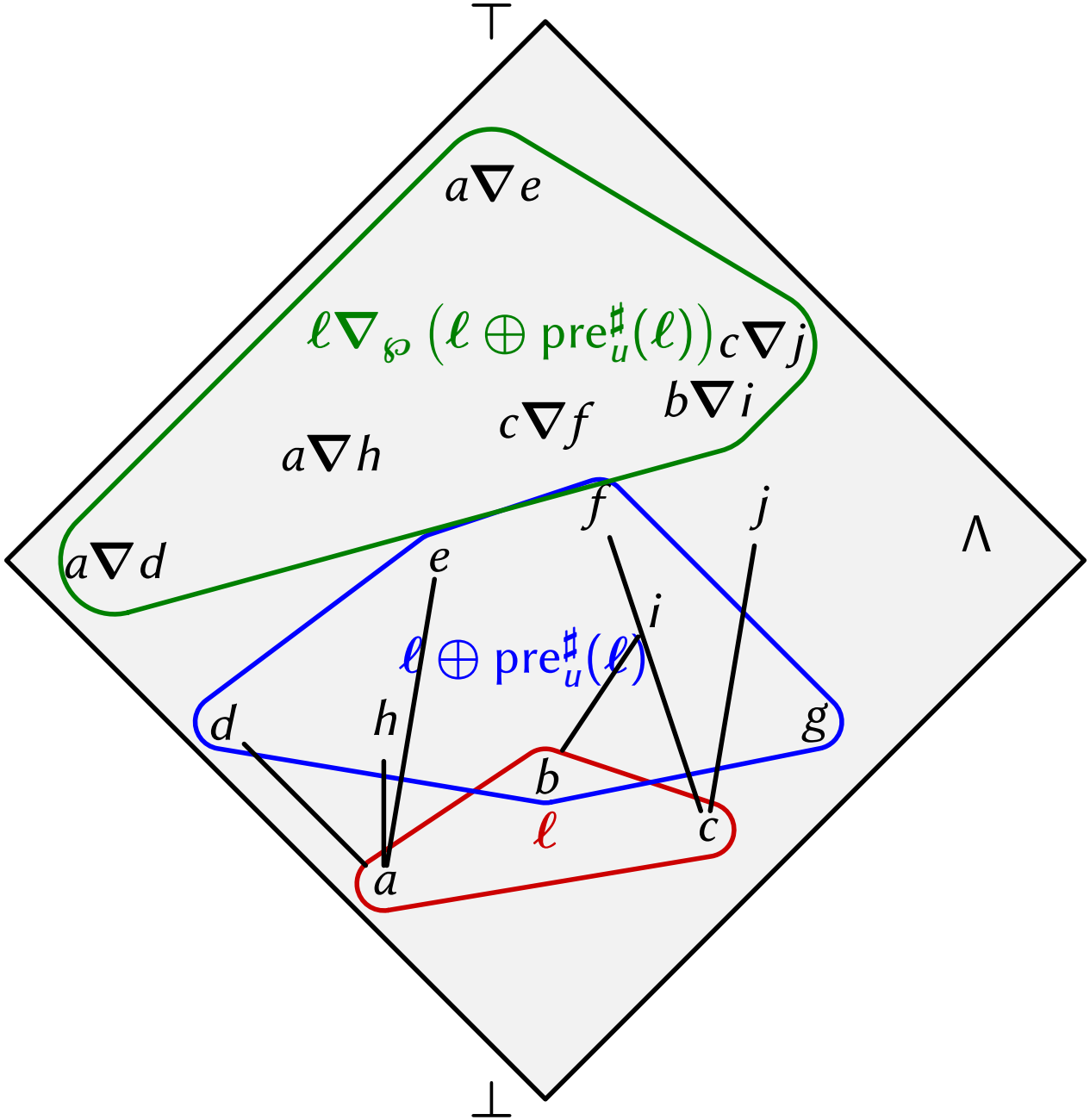
Egli-Milner Ordering Enforcement for Widening in $\wp(\Lambda)$



Egli-Milner Ordering Enforcement for Widening in $\wp(\Lambda)$



Egli-Milner Ordering Enforcement for Widening in $\wp(\Lambda)$



Handling Disjunctions & Avoiding Deadlocks: Basic Idea

Powerset Domain Extension

- Handling of Disjunctions for Non-convex *Bad*
- Enforcing Reactivity of the Result

Proposals

1. Previous Algorithm with Powerset Extension
2. Custom, Double Fixpoint Algorithm

Handling Disjunctions & Avoiding Deadlocks: Basic Idea

Powerset Domain Extension

- Handling of Disjunctions for Non-convex *Bad*
- Enforcing Reactivity of the Result

Proposals

1. Previous Algorithm with Powerset Extension
   - Problem with Partitioned CFGs (Sink Locations)
2. Custom, Double Fixpoint Algorithm

Handling Disjunctions & Avoiding Deadlocks: Basic Idea

Powerset Domain Extension

- Handling of Disjunctions for Non-convex *Bad*
- Enforcing Reactivity of the Result

Proposals

1. Previous Algorithm with Powerset Extension
   - Problem with Partitioned CFGs (Sink Locations)
2. Custom, Double Fixpoint Algorithm

$$\text{coreach}_u^{\wp}(B) = \text{lfp}\left(\lambda\beta.B\boldsymbol{\nabla}_{\wp}\ldots\text{pre}_u^{\wp}\left(\text{coreach}_u^{\boldsymbol{\nabla}}(\beta)\right)\right)$$

Outline

ReaX

- Extension of ReaVer[3]
- Combining *Decision Diagrams* and *Numerical Abstract Domains* with BddApron[4]
  - Numerical Abstract Domains: APRON[5] (Boxes, Convex Polyhedra…)
  - Decision Diagrams: CUDD
- Heptagon/BZR Backend & Frontend
- Triangulation of the Controller
- One-step Optimization of Numerical State Variables

---

[3] Peter Schrammel. "Logico-Numerical Verification Methods for Discrete and Hybrid Systems". PhD thesis. University of Grenoble, 2012.

[4] Bertrand Jeannet. *BddApron: A logico-numerical abstract domain library*. 2009. URL: http://pop-art.inrialpes.fr/~bjeannet/bjeannet-forge/bddapron/.

[5] Bertrand Jeannet and Antoine Miné. "APRON: A Library of Numerical Abstract Domains for Static Analysis". In: *Proceedings of the 21st International Conference on Computer Aided Verification*. CAV '09. Grenoble, France: Springer-Verlag, 2009, pp. 661–667. URL: http://apron.cri.ensmp.fr/library/.

Outline

Conclusion & Further Works

Overall Contributions

- Algorithms for the Safety Control Problem of Infinite State Systems
  - Using Abstract Interpretation Techniques
  - Handling Disjunctions
  - Reactivity-aware
- Implementation in ReaX
  - Efficient Synthesis in the Finite Case
  - Heptagon/BZR Backend & Frontend $\rightsquigarrow$ Favorably Replaces SIGALI

Conclusion & Further Works

Overall Contributions

- Algorithms for the Safety Control Problem of Infinite State Systems
  - Using Abstract Interpretation Techniques
  - Handling Disjunctions
  - Reactivity-aware
- Implementation in ReaX
  - Efficient Synthesis in the Finite Case
  - Heptagon/BZR Backend & Frontend $\rightsquigarrow$ Favorably Replaces SIGALI

Forthcoming Challenges

- Synthesis Failure Diagnosis
- Improving Precision
  - Dynamic Partitioning
  - Abstract Acceleration (Avoids Some Widenings)

Outline

- ASTS Model of Logico-numerical Reactive Programs
- Safety Control Problem for ASTSs
- Discrete Controller Synthesis for ASTSs
- Some Limitations & Solutions
- ReaX
- Conclusions

Outline

Outline

Example Execution of Deadlock-avoiding Algorithm

Invariant: $\Phi\left(\langle x, y\rangle\right) = \left(v_{x<2} \wedge v_{y<2}\right)$ $\rightsquigarrow Bad = (\neg v_{x<2} \vee \neg v_{y<2})$

$\ell_0 = \alpha_{\wp}(\neg v_{x<2} \vee \neg v_{y<2}) = \{x \geqslant 2; y \geqslant 2\}$

Example Execution of Deadlock-avoiding Algorithm

Invariant: $\Phi\left(\langle x, y\rangle\right) = \left(v_{x<2} \wedge v_{y<2}\right)$ $\rightsquigarrow Bad = \left(\neg v_{x<2} \vee \neg v_{y<2}\right)$

$\ell_0 = \alpha_{\wp}(\neg v_{x<2} \vee \neg v_{y<2}) = \{x \geqslant 2; y \geqslant 2\}$

$\ell_\infty = \text{coreach}_u^{\nabla}(\ell_0) = \text{lfp}\left(\lambda\beta.\ell_0 \nabla_{\wp}\left(\beta \sqcup \text{pre}_u^{\sharp}(\beta)\right)\right)$

Example Execution of Deadlock-avoiding Algorithm

Invariant: $\Phi\left(\langle x, y\rangle\right) = \left(v_{x<2} \wedge v_{y<2}\right)$ $\rightsquigarrow Bad = \left(\neg v_{x<2} \vee \neg v_{y<2}\right)$

$\ell_0 = \alpha_{\wp}(\neg v_{x<2} \vee \neg v_{y<2}) = \{x \geqslant 2; y \geqslant 2\}$

$\ell_\infty = \text{coreach}_u^{\boldsymbol{\nabla}}(\ell_0) = \text{lfp}\left(\lambda\beta.\ell_0 \boldsymbol{\nabla}_{\wp}\left(\beta \sqcup \text{pre}_u^\sharp(\beta)\right)\right)$

- Let $\ell = \text{pre}_u^\sharp(\ell_0) =$

Example Execution of Deadlock-avoiding Algorithm

Invariant: $\Phi\left(\langle x, y\rangle\right) = \left(v_{x<2} \wedge v_{y<2}\right)$ $\rightsquigarrow Bad = \left(\neg v_{x<2} \vee \neg v_{y<2}\right)$

$\ell_0 = \alpha_{\wp}(\neg v_{x<2} \vee \neg v_{y<2}) = \{x \geqslant 2; y \geqslant 2\}$

$\ell_\infty = \text{coreach}_u^{\boldsymbol{\nabla}}(\ell_0) = \text{lfp}\left(\lambda\beta.\ell_0 \boldsymbol{\nabla}_{\wp} \left(\beta \sqcup \text{pre}_u^\sharp(\beta)\right)\right)$

- Let $\ell = \text{pre}_u^\sharp(\ell_0) = \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_{\varnothing}^{\wp}, T^{\sharp -1}\left(\ell_0\right)\right)\right)$

Example Execution of Deadlock-avoiding Algorithm

Invariant: $\Phi\left(\langle x, y\rangle\right) = \left(v_{x<2} \wedge v_{y<2}\right)$ $\rightsquigarrow Bad = \left(\neg v_{x<2} \vee \neg v_{y<2}\right)$

$\ell_0 = \alpha_{\wp}(\neg v_{x<2} \vee \neg v_{y<2}) = \{x \geqslant 2; y \geqslant 2\}$

$\ell_\infty = \text{coreach}_u^{\boldsymbol{\nabla}}(\ell_0) = \text{lfp}\left(\lambda\beta.\ell_0 \boldsymbol{\nabla}_{\wp}\left(\beta \sqcup \text{pre}_u^\sharp(\beta)\right)\right)$

- Let $\ell = \text{pre}_u^\sharp(\ell_0) = \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_{\varnothing}^{\wp}, T^{\sharp -1}\left(\ell_0\right)\right)\right)$
  - Let $\ell' = T^{\sharp -1}\left(\ell_0\right)$

Example Execution of Deadlock-avoiding Algorithm

Invariant: $\Phi\left(\langle x, y\rangle\right) = \left(v_{x<2} \wedge v_{y<2}\right)$ $\rightsquigarrow Bad = \left(\neg v_{x<2} \vee \neg v_{y<2}\right)$

$\ell_0 = \alpha_{\wp}(\neg v_{x<2} \vee \neg v_{y<2}) = \{x \geqslant 2; y \geqslant 2\}$

$\ell_\infty = \text{coreach}_u^{\boldsymbol{\nabla}}(\ell_0) = \text{lfp}\left(\lambda\beta.\ell_0 \boldsymbol{\nabla}_{\wp}\left(\beta \sqcup \text{pre}_u^\sharp(\beta)\right)\right)$

- Let $\ell = \text{pre}_u^\sharp(\ell_0) = \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_{\varnothing}^{\wp}, T^{\sharp -1}\left(\ell_0\right)\right)\right)$
  - Let $\ell' = \{(c \wedge x \geqslant 1 \vee \neg c \wedge y \geqslant 2); (c \wedge y \geqslant 2 \vee \neg c \wedge y \geqslant 1)\}$

Example Execution of Deadlock-avoiding Algorithm

Invariant: $\Phi\left(\langle x, y\rangle\right) = \left(v_{x<2} \wedge v_{y<2}\right)$ $\rightsquigarrow Bad = \left(\neg v_{x<2} \vee \neg v_{y<2}\right)$

$\ell_0 = \alpha_{\wp}(\neg v_{x<2} \vee \neg v_{y<2}) = \{x \geqslant 2; y \geqslant 2\}$

$\ell_\infty = \text{coreach}_u^{\boldsymbol{\nabla}}(\ell_0) = \text{lfp}\left(\lambda\beta.\ell_0 \boldsymbol{\nabla}_{\wp} \left(\beta \sqcup \text{pre}_u^\sharp(\beta)\right)\right)$

- Let $\ell = \text{pre}_u^\sharp(\ell_0) = \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_\varnothing^{\wp}, T^{\sharp -1}\left(\ell_0\right)\right)\right)$
  - Let $\ell' = \{(c \wedge x \geqslant 1 \vee \neg c \wedge y \geqslant 2); (c \wedge y \geqslant 2 \vee \neg c \wedge y \geqslant 1)\}$
  - $\alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_\varnothing^{\wp}, \ell'\right)\right)$

Example Execution of Deadlock-avoiding Algorithm

Invariant: $\Phi\left(\langle x, y\rangle\right) = \left(v_{x<2} \wedge v_{y<2}\right)$ $\rightsquigarrow Bad = \left(\neg v_{x<2} \vee \neg v_{y<2}\right)$

$\ell_0 = \alpha_{\wp}(\neg v_{x<2} \vee \neg v_{y<2}) = \{x \geqslant 2; y \geqslant 2\}$

$\ell_\infty = \text{coreach}_u^{\boldsymbol{\nabla}}(\ell_0) = \text{lfp}\left(\lambda\beta.\ell_0 \boldsymbol{\nabla}_{\wp} \left(\beta \sqcup \text{pre}_u^\sharp(\beta)\right)\right)$

- Let $\ell = \text{pre}_u^\sharp(\ell_0) = \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_\varnothing^{\wp}, T^{\sharp -1}\left(\ell_0\right)\right)\right)$
  - Let $\ell' = \{(c \wedge x \geqslant 1 \vee \neg c \wedge y \geqslant 2); (c \wedge y \geqslant 2 \vee \neg c \wedge y \geqslant 1)\}$
  - $\alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_\varnothing^{\wp}, \ell'\right)\right)$
    $= \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\left(c \wedge \left(v_{x\geqslant 1} \vee v_{y\geqslant 2}\right)\right) \vee \left(\neg c \wedge \left(v_{x\geqslant 2} \vee v_{y\geqslant 1}\right)\right)\right)\right)$

Example Execution of Deadlock-avoiding Algorithm

Invariant: $\Phi\left(\langle x, y\rangle\right) = \left(v_{x<2} \wedge v_{y<2}\right)$ $\rightsquigarrow Bad = \left(\neg v_{x<2} \vee \neg v_{y<2}\right)$

$\ell_0 = \alpha_{\wp}(\neg v_{x<2} \vee \neg v_{y<2}) = \{x \geqslant 2; y \geqslant 2\}$

$\ell_\infty = \text{coreach}_u^{\boldsymbol{\nabla}}(\ell_0) = \text{lfp}\left(\lambda\beta.\ell_0 \boldsymbol{\nabla}_{\wp} \left(\beta \sqcup \text{pre}_u^\sharp(\beta)\right)\right)$

- Let $\ell = \text{pre}_u^\sharp(\ell_0) = \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_\varnothing^{\wp}, T^{\sharp -1}\left(\ell_0\right)\right)\right)$
  - Let $\ell' = \{(c \wedge x \geqslant 1 \vee \neg c \wedge y \geqslant 2); (c \wedge y \geqslant 2 \vee \neg c \wedge y \geqslant 1)\}$
  - $\alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_\varnothing^{\wp}, \ell'\right)\right)$
    $= \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\left(c \wedge \left(v_{x\geqslant 1} \vee v_{y\geqslant 2}\right)\right) \vee \left(\neg c \wedge \left(v_{x\geqslant 2} \vee v_{y\geqslant 1}\right)\right)\right)\right)$
    $= \alpha_{\wp}\left(\left(v_{x\geqslant 1} \vee v_{y\geqslant 2}\right) \wedge \left(v_{x\geqslant 2} \vee v_{y\geqslant 1}\right)\right)$

Example Execution of Deadlock-avoiding Algorithm

Invariant: $\Phi\left(\langle x, y\rangle\right) = \left(v_{x<2} \wedge v_{y<2}\right)$ $\rightsquigarrow Bad = \left(\neg v_{x<2} \vee \neg v_{y<2}\right)$

$\ell_0 = \alpha_{\wp}(\neg v_{x<2} \vee \neg v_{y<2}) = \{x \geqslant 2; y \geqslant 2\}$

$\ell_\infty = \text{coreach}_u^{\boldsymbol{\nabla}}(\ell_0) = \text{lfp}\left(\lambda\beta.\ell_0 \boldsymbol{\nabla}_{\wp} \left(\beta \sqcup \text{pre}_u^\sharp(\beta)\right)\right)$

- Let $\ell = \text{pre}_u^\sharp(\ell_0) = \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_\varnothing^{\wp}, T^{\sharp -1}\left(\ell_0\right)\right)\right)$
  - Let $\ell' = \{(c \wedge x \geqslant 1 \vee \neg c \wedge y \geqslant 2); (c \wedge y \geqslant 2 \vee \neg c \wedge y \geqslant 1)\}$
  - $\alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_\varnothing^{\wp}, \ell'\right)\right)$
    $= \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\left(c \wedge \left(v_{x\geqslant 1} \vee v_{y\geqslant 2}\right)\right) \vee \left(\neg c \wedge \left(v_{x\geqslant 2} \vee v_{y\geqslant 1}\right)\right)\right)\right)$
    $= \alpha_{\wp}\left(\left(v_{x\geqslant 1} \vee v_{y\geqslant 2}\right) \wedge \left(v_{x\geqslant 2} \vee v_{y\geqslant 1}\right)\right)$
    $= \{x \geqslant 2; x \geqslant 1 \wedge y \geqslant 1; x \geqslant 2 \wedge y \geqslant 2; y \geqslant 2\}$

Example Execution of Deadlock-avoiding Algorithm

Invariant: $\Phi\left(\langle x, y\rangle\right) = \left(v_{x<2} \wedge v_{y<2}\right)$ $\rightsquigarrow Bad = \left(\neg v_{x<2} \vee \neg v_{y<2}\right)$

$\ell_0 = \alpha_{\wp}(\neg v_{x<2} \vee \neg v_{y<2}) = \{x \geqslant 2; y \geqslant 2\}$

$\ell_\infty = \text{coreach}_u^{\boldsymbol{\nabla}}(\ell_0) = \text{lfp}\left(\lambda\beta.\ell_0 \boldsymbol{\nabla}_{\wp}\left(\beta \sqcup \text{pre}_u^\sharp(\beta)\right)\right)$

- Let $\ell = \text{pre}_u^\sharp(\ell_0) = \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_\varnothing^{\wp}, T^{\sharp -1}\left(\ell_0\right)\right)\right)$
  - Let $\ell' = \{(c \wedge x \geqslant 1 \vee \neg c \wedge y \geqslant 2); (c \wedge y \geqslant 2 \vee \neg c \wedge y \geqslant 1)\}$
  - $\alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_\varnothing^{\wp}, \ell'\right)\right)$
    $= \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\left(c \wedge \left(v_{x\geqslant 1} \vee v_{y\geqslant 2}\right)\right) \vee \left(\neg c \wedge \left(v_{x\geqslant 2} \vee v_{y\geqslant 1}\right)\right)\right)\right)$
    $= \alpha_{\wp}\left(\left(v_{x\geqslant 1} \vee v_{y\geqslant 2}\right) \wedge \left(v_{x\geqslant 2} \vee v_{y\geqslant 1}\right)\right)$
    $= \{x \geqslant 2; x \geqslant 1 \wedge y \geqslant 1; y \geqslant 2\}$

Example Execution of Deadlock-avoiding Algorithm

Invariant: $\Phi\left(\langle x, y\rangle\right) = \left(v_{x<2} \wedge v_{y<2}\right)$ $\rightsquigarrow Bad = \left(\neg v_{x<2} \vee \neg v_{y<2}\right)$

$\ell_0 = \alpha_{\wp}(\neg v_{x<2} \vee \neg v_{y<2}) = \{x \geqslant 2; y \geqslant 2\}$

$\ell_\infty = \text{coreach}_u^{\nabla}(\ell_0) = \text{lfp}\left(\lambda\beta.\ell_0 \nabla_{\wp}\left(\beta \sqcup \text{pre}_u^{\sharp}(\beta)\right)\right)$

- Let $\ell = \text{pre}_u^{\sharp}(\ell_0) = \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_{\varnothing}^{\wp}, T^{\sharp -1}\left(\ell_0\right)\right)\right)$
  - Let $\ell' = \{(c \wedge x \geqslant 1 \vee \neg c \wedge y \geqslant 2); (c \wedge y \geqslant 2 \vee \neg c \wedge y \geqslant 1)\}$
  - $\alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_{\varnothing}^{\wp}, \ell'\right)\right)$
    $= \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\left(c \wedge \left(v_{x\geqslant 1} \vee v_{y\geqslant 2}\right)\right) \vee \left(\neg c \wedge \left(v_{x\geqslant 2} \vee v_{y\geqslant 1}\right)\right)\right)\right)$
    $= \alpha_{\wp}\left(\left(v_{x\geqslant 1} \vee v_{y\geqslant 2}\right) \wedge \left(v_{x\geqslant 2} \vee v_{y\geqslant 1}\right)\right)$
    $= \{x \geqslant 2; x \geqslant 1 \wedge y \geqslant 1; y \geqslant 2\}$
- $\left(\ell_0 \nabla_{\wp}\left(\ell_0 \oplus \ell\right)\right) = \ell_0 \nabla_{\wp}\{x \geqslant 2; y \geqslant 2; x \geqslant 1 \wedge y \geqslant 1\}$

Example Execution of Deadlock-avoiding Algorithm

Invariant: $\Phi(\langle x, y\rangle) = (v_{x<2} \wedge v_{y<2}) \rightsquigarrow \mathit{Bad} = (\neg v_{x<2} \vee \neg v_{y<2})$

$\ell_0 = \alpha_{\wp}(\neg v_{x<2} \vee \neg v_{y<2}) = \{x \geqslant 2; y \geqslant 2\}$

$\ell_\infty = \mathrm{coreach}_u^{\boldsymbol{\nabla}}(\ell_0) = \mathrm{lfp}\left(\lambda\beta.\ell_0 \boldsymbol{\nabla}_{\wp}\left(\beta \sqcup \mathrm{pre}_u^\sharp(\beta)\right)\right)$

- Let $\ell = \mathrm{pre}_u^\sharp(\ell_0) = \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_{\varnothing}^{\wp}, T^{\sharp -1}(\ell_0)\right)\right)$
  - Let $\ell' = \{(c \wedge x \geqslant 1 \vee \neg c \wedge y \geqslant 2); (c \wedge y \geqslant 2 \vee \neg c \wedge y \geqslant 1)\}$
  - $\alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_{\varnothing}^{\wp}, \ell'\right)\right)$
    $= \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\left(c \wedge (v_{x\geqslant 1} \vee v_{y\geqslant 2})\right) \vee \left(\neg c \wedge (v_{x\geqslant 2} \vee v_{y\geqslant 1})\right)\right)\right)$
    $= \alpha_{\wp}\left((v_{x\geqslant 1} \vee v_{y\geqslant 2}) \wedge (v_{x\geqslant 2} \vee v_{y\geqslant 1})\right)$
    $= \{x \geqslant 2; x \geqslant 1 \wedge y \geqslant 1; y \geqslant 2\}$
- $(\ell_0 \boldsymbol{\nabla}_{\wp}(\ell_0 \oplus \ell)) = \ell_0 \boldsymbol{\nabla}_{\wp}\{x \geqslant 2; y \geqslant 2; x \geqslant 1 \wedge y \geqslant 1\}$
  - Egli-Milner Ordering Enforcement
    - $x \geqslant 2 \sqcup (x \geqslant 1 \wedge y \geqslant 1) = x \geqslant 1$
    - $y \geqslant 2 \sqcup (x \geqslant 1 \wedge y \geqslant 1) = y \geqslant 1$

Example Execution of Deadlock-avoiding Algorithm

Invariant: $\Phi\left(\langle x, y\rangle\right) = \left(v_{x<2} \wedge v_{y<2}\right)$ $\rightsquigarrow Bad = \left(\neg v_{x<2} \vee \neg v_{y<2}\right)$

$\ell_0 = \alpha_{\wp}(\neg v_{x<2} \vee \neg v_{y<2}) = \{x \geqslant 2; y \geqslant 2\}$

$\ell_\infty = \text{coreach}_u^{\nabla}(\ell_0) = \text{lfp}\left(\lambda\beta.\ell_0 \nabla_{\wp}\left(\beta \sqcup \text{pre}_u^{\sharp}(\beta)\right)\right)$

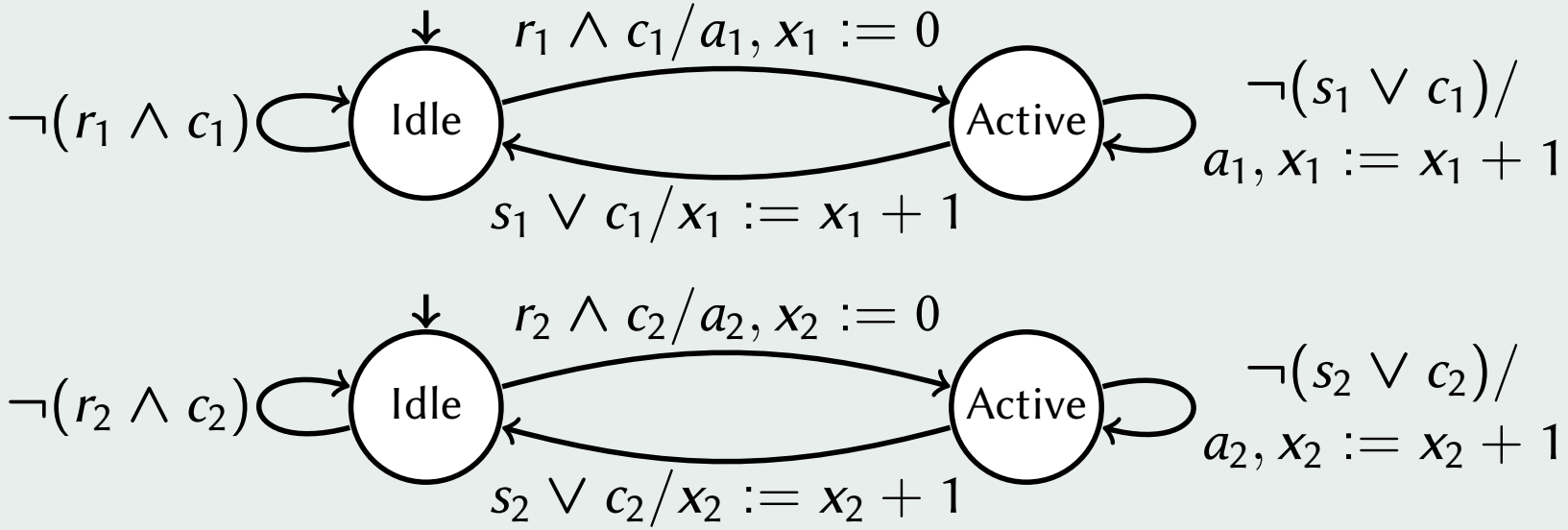
- Let $\ell = \text{pre}_u^{\sharp}(\ell_0) = \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_{\varnothing}^{\wp}, T^{\sharp -1}\left(\ell_0\right)\right)\right)$
  - Let $\ell' = \{(c \wedge x \geqslant 1 \vee \neg c \wedge y \geqslant 2); (c \wedge y \geqslant 2 \vee \neg c \wedge y \geqslant 1)\}$
  - $\alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\exists_{\varnothing}^{\wp}, \ell'\right)\right)$
    $= \alpha_{\wp}\left(\forall_C, \gamma_{\wp}\left(\left(c \wedge \left(v_{x\geqslant 1} \vee v_{y\geqslant 2}\right)\right) \vee \left(\neg c \wedge \left(v_{x\geqslant 2} \vee v_{y\geqslant 1}\right)\right)\right)\right)$
    $= \alpha_{\wp}\left(\left(v_{x\geqslant 1} \vee v_{y\geqslant 2}\right) \wedge \left(v_{x\geqslant 2} \vee v_{y\geqslant 1}\right)\right)$
    $= \{x \geqslant 2; x \geqslant 1 \wedge y \geqslant 1; y \geqslant 2\}$
- $\left(\ell_0 \nabla_{\wp}\left(\ell_0 \oplus \ell\right)\right) = \ell_0 \nabla_{\wp} \{x \geqslant 2; y \geqslant 2; x \geqslant 1 \wedge y \geqslant 1\}$
  - Egli-Milner Ordering Enforcement
    - $x \geqslant 2 \sqcup (x \geqslant 1 \wedge y \geqslant 1) = x \geqslant 1$
    - $y \geqslant 2 \sqcup (x \geqslant 1 \wedge y \geqslant 1) = y \geqslant 1$

$= \{x \geqslant 2 \nabla x \geqslant 1; y \geqslant 2 \nabla y \geqslant 1\} = \top_{\wp}$

Outline

Example Safety Control Problems for Infinite System

Example Infinite State System \ Invariants

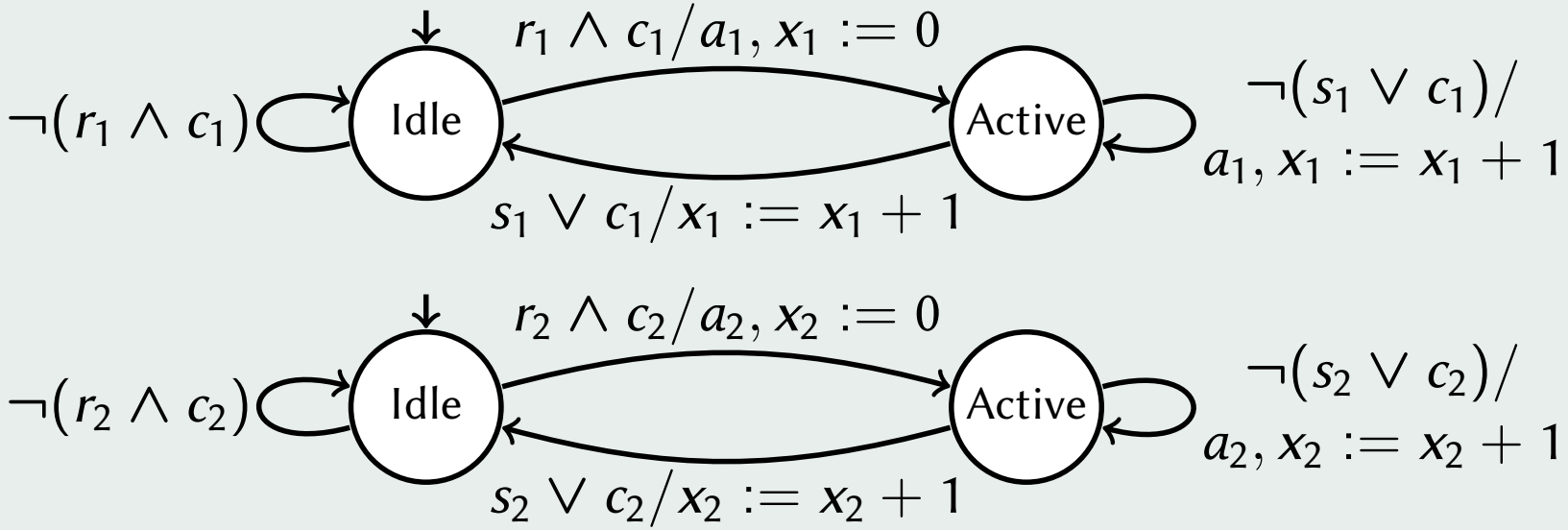


- $X = \langle t_1, t_2, x_1, x_2, a_1, a_2 \rangle$ $\quad \mathcal{D}_X = \{\text{Idle}, \text{Active}\}^2 \times \mathbb{Z}^2 \times \mathbb{B}^2$
- $I_{uc} = \langle r_1, r_2, s_1, s_2 \rangle, I_c = \langle c_1, c_2 \rangle$ $\quad \mathcal{D}_{I_{uc}} = \mathbb{B}^4, \mathcal{D}_{I_c} = \mathbb{B}^2$
- Enforcing Mutual Exclusion Between Active States

$$\Phi(\langle t_1, t_2, x_1, x_2 \ldots \rangle) = (t_1 \neq t_2 \vee t_1 = \text{Idle})$$

Example Safety Control Problems for Infinite System

Example Infinite State System \ Invariants

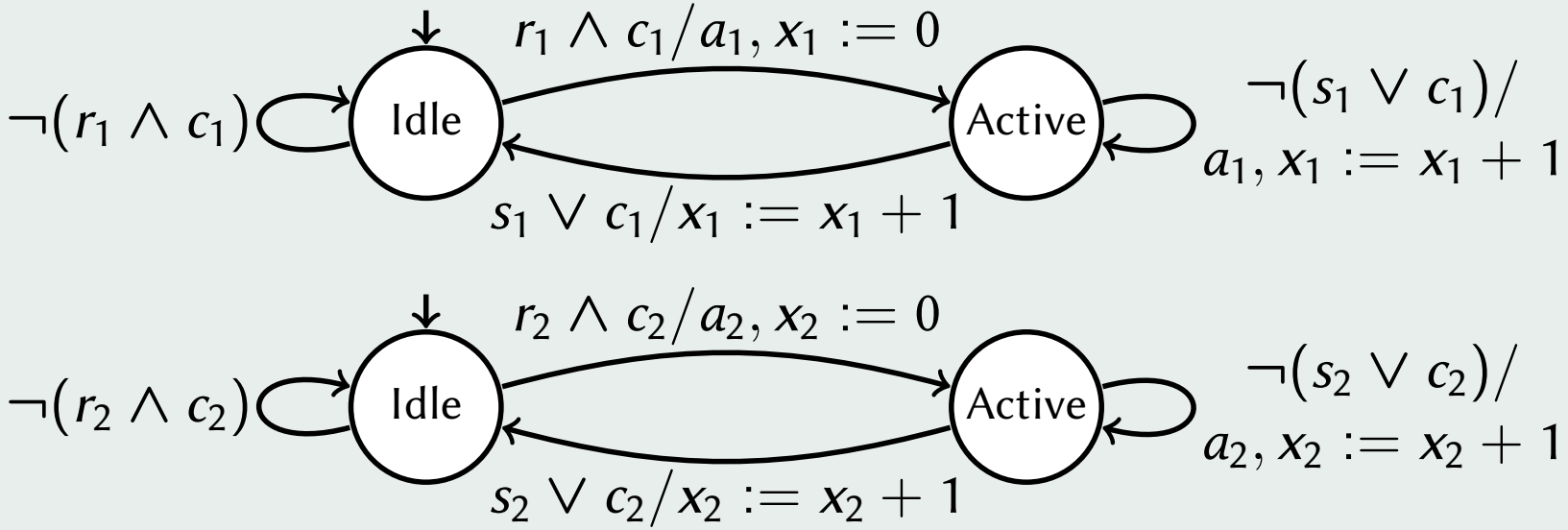


- $X = \langle t_1, t_2, x_1, x_2, a_1, a_2 \rangle$ $\quad \mathcal{D}_X = \{\text{Idle}, \text{Active}\}^2 \times \mathbb{Z}^2 \times \mathbb{B}^2$
- $I_{uc} = \langle r_1, r_2, s_1, s_2 \rangle, I_c = \langle c_1, c_2 \rangle$ $\quad \mathcal{D}_{I_{uc}} = \mathbb{B}^4, \mathcal{D}_{I_c} = \mathbb{B}^2$
- Enforcing Mutual Exclusion Between Active States & Bounds on x_i's

$$\Phi(\langle t_1, t_2, x_1, x_2 \ldots \rangle) = (t_1 \neq t_2 \vee t_1 = \text{Idle}) \wedge (x_1 \leqslant 10 \wedge x_2 \leqslant 10)$$

Example Safety Control Problems for Infinite System

Example Infinite State System \ Invariants

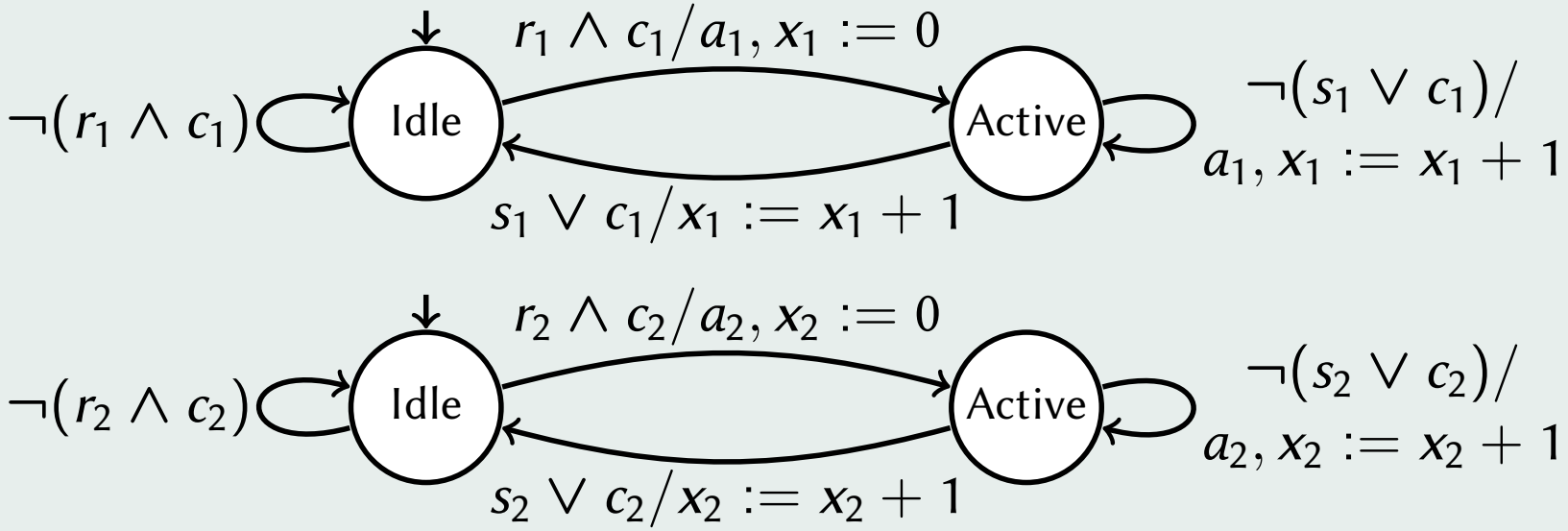


- $X = \langle t_1, t_2, x_1, x_2, a_1, a_2 \rangle$ $\quad \mathcal{D}_X = \{\text{Idle}, \text{Active}\}^2 \times \mathbb{Z}^2 \times \mathbb{B}^2$
- $I_{uc} = \langle r_1, r_2, s_1, s_2 \rangle, I_c = \langle c_1, c_2 \rangle$ $\quad \mathcal{D}_{I_{uc}} = \mathbb{B}^4, \mathcal{D}_{I_c} = \mathbb{B}^2$
- Enforcing Mutual Exclusion Between Active States & Bounds on x_i's & Relational Constraints on x_i's

$$\Phi(\langle t_1, t_2, x_1, x_2 \ldots \rangle) = (t_1 \neq t_2 \vee t_1 = \text{Idle}) \wedge (x_1 \leqslant 10 \wedge x_2 \leqslant 10) \wedge (x_1 \leqslant x_2)$$

Results of Over-approximating Algorithm

System



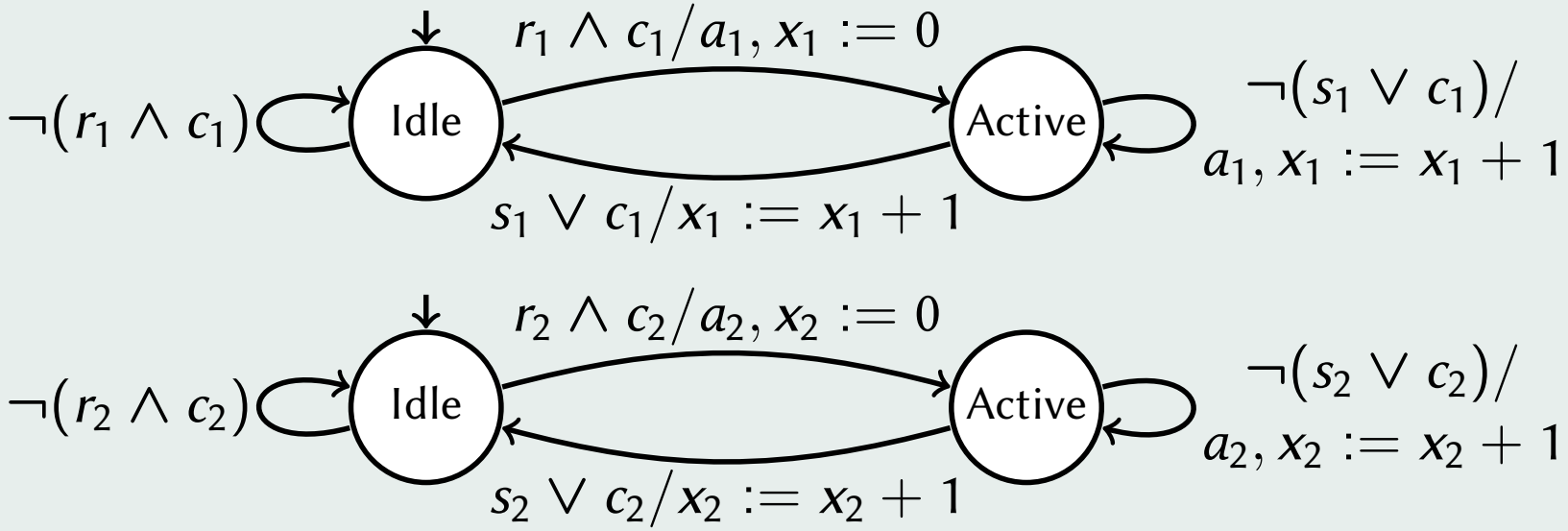
$\Phi(\langle t_1, t_2, x_1 \ldots \rangle) = (t_1 \neq t_2 \vee t_1 = \text{Idle}) \wedge (x_1 \leqslant 10 \wedge x_2 \leqslant 10) \wedge (x_1 \leqslant x_2)$

- Power Domain, Intervals

$$\begin{aligned} I'_{Bad} &\supseteq \{t_1 = \text{Idle} \wedge t_2 = \text{Idle} \wedge x_1 < 11 \wedge x_2 < 10\} \\ &\supseteq \{t_1 = \text{Idle} \wedge t_2 = \text{Idle} \wedge x_1 = 0 \wedge x_2 = 0\} = \mathcal{X}_0 \end{aligned}$$

Results of Over-approximating Algorithm

System



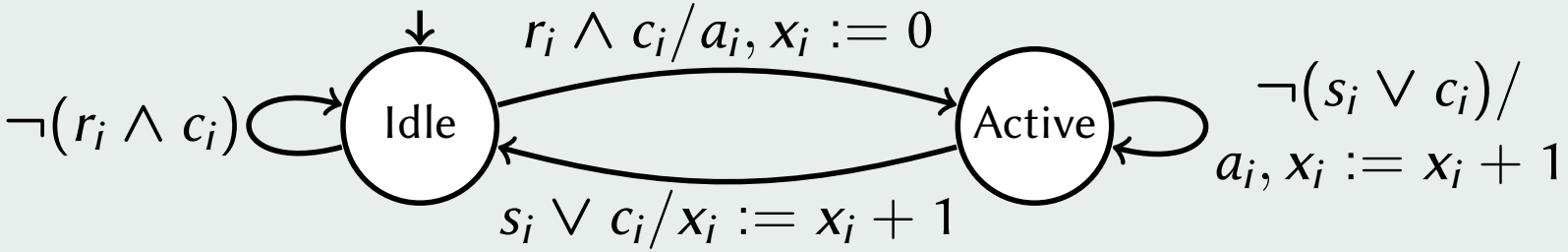
$\Phi(\langle t_1, t_2, x_1 \ldots \rangle) = (t_1 \neq t_2 \vee t_1 = \text{Idle}) \wedge (x_1 \leqslant 10 \wedge x_2 \leqslant 10) \wedge (x_1 \leqslant x_2)$

- Power Domain, Convex Polyhedra

$$I'^{c}_{Bad} = \left\{ \begin{array}{l} (t_1 = \text{Idle} \wedge t_2 = \text{Idle} \wedge x_1 \leqslant x_2 \leqslant 10) \vee \\ (t_1 = \text{Idle} \wedge t_2 = \text{Active} \wedge x_1 \leqslant x_2 \leqslant 9) \vee \\ (t_1 = \text{Active} \wedge t_2 = \text{Idle} \wedge x_1 < x_2 \leqslant 10) \end{array} \right\}$$

Performance Evaluation

Example Infinite State System \ Invariants



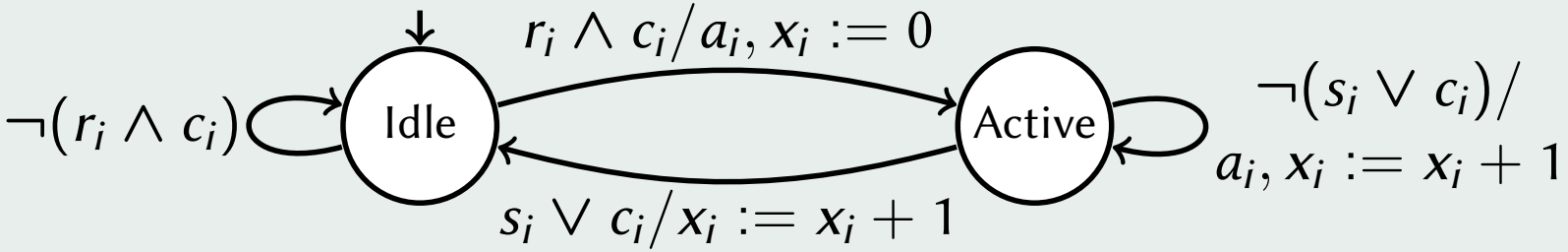
- $X = \langle t_1 \dots t_n, x_1 \dots x_n, a_1 \dots a_n \rangle$ $\quad \mathcal{D}_X = \{\text{Idle}, \text{Active}\}^n \times \mathbb{Z}^n \times \mathbb{B}^n$
- $I_{uc} = \langle r_1 \dots r_n, s_1 \dots s_n \rangle, I_c = \langle c_1 \dots c_n \rangle$ $\quad \mathcal{D}_{I_{uc}} = \mathbb{B}^{2n}, \mathcal{D}_{I_c} = \mathbb{B}^n$
- Enforcing Mutual Exclusion Between Active States & Bounds on x_i's

$$\Phi(\langle t_1 \dots t_n, x_1 \dots x_n, \dots \rangle) =$$

$$\bigoplus_{i \in [1,n]} (t_i = \text{Active}) \wedge \bigwedge_{i \in [1,n]} (x_i \leqslant 10)$$

Performance Evaluation

Example Infinite State System \ Invariants



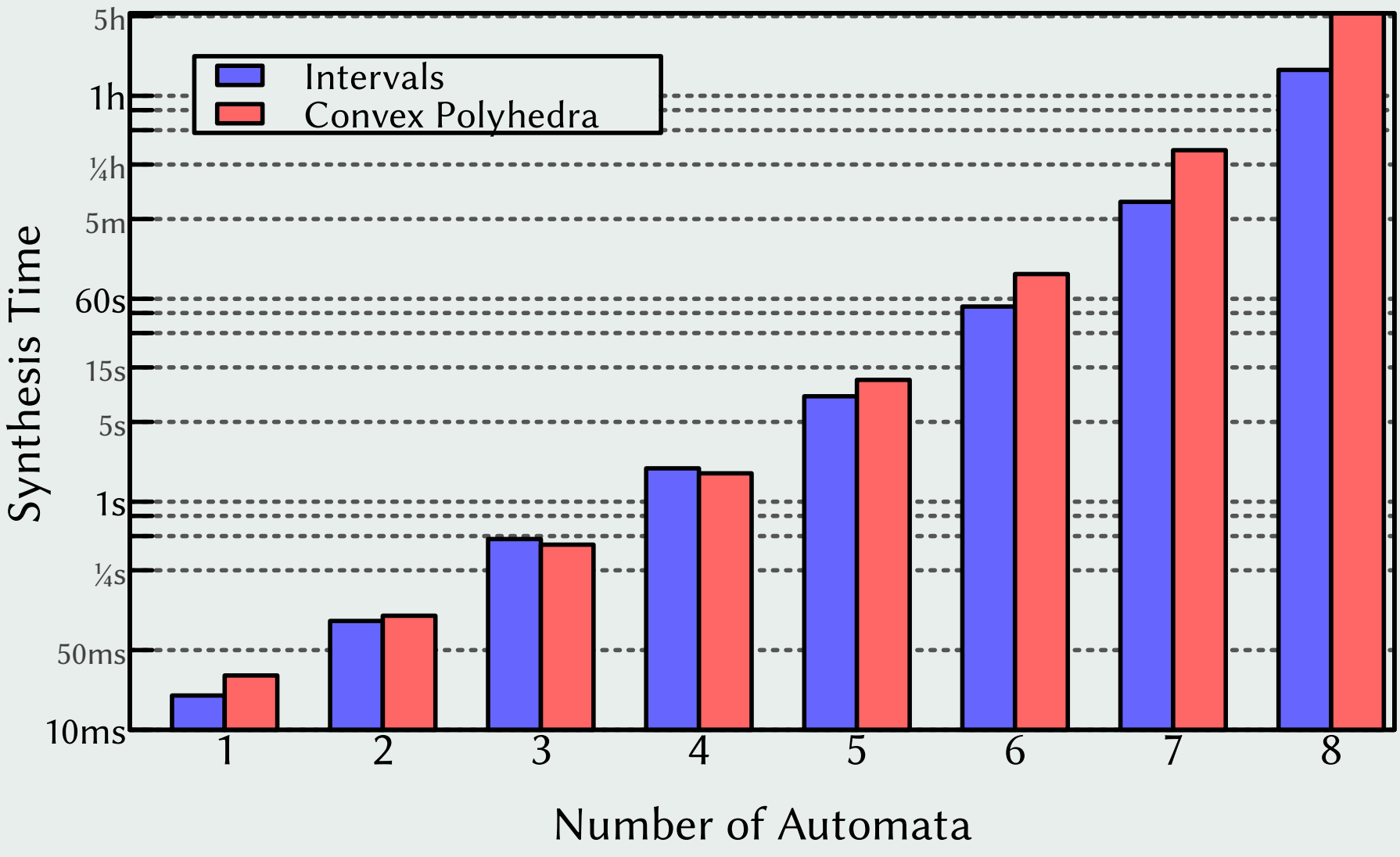
- $X = \langle t_1 \dots t_n, x_1 \dots x_n, a_1 \dots a_n \rangle$ $\quad \mathcal{D}_X = \{\text{Idle}, \text{Active}\}^n \times \mathbb{Z}^n \times \mathbb{B}^n$
- $I_{uc} = \langle r_1 \dots r_n, s_1 \dots s_n \rangle, I_c = \langle c_1 \dots c_n \rangle$ $\quad \mathcal{D}_{I_{uc}} = \mathbb{B}^{2n}, \mathcal{D}_{I_c} = \mathbb{B}^n$
- Enforcing Mutual Exclusion Between Active States & Bounds on x_i's & Relational Constraints on x_i's

$$\Phi(\langle t_1 \dots t_n, x_1 \dots x_n, \dots \rangle) =$$

$$\bigoplus_{i \in [1,n]} (t_i = \text{Active}) \wedge \bigwedge_{i \in [1,n]} (x_i \leqslant 10) \wedge \bigwedge_{i \in [1,n[} (x_i \leqslant x_{i+1})$$

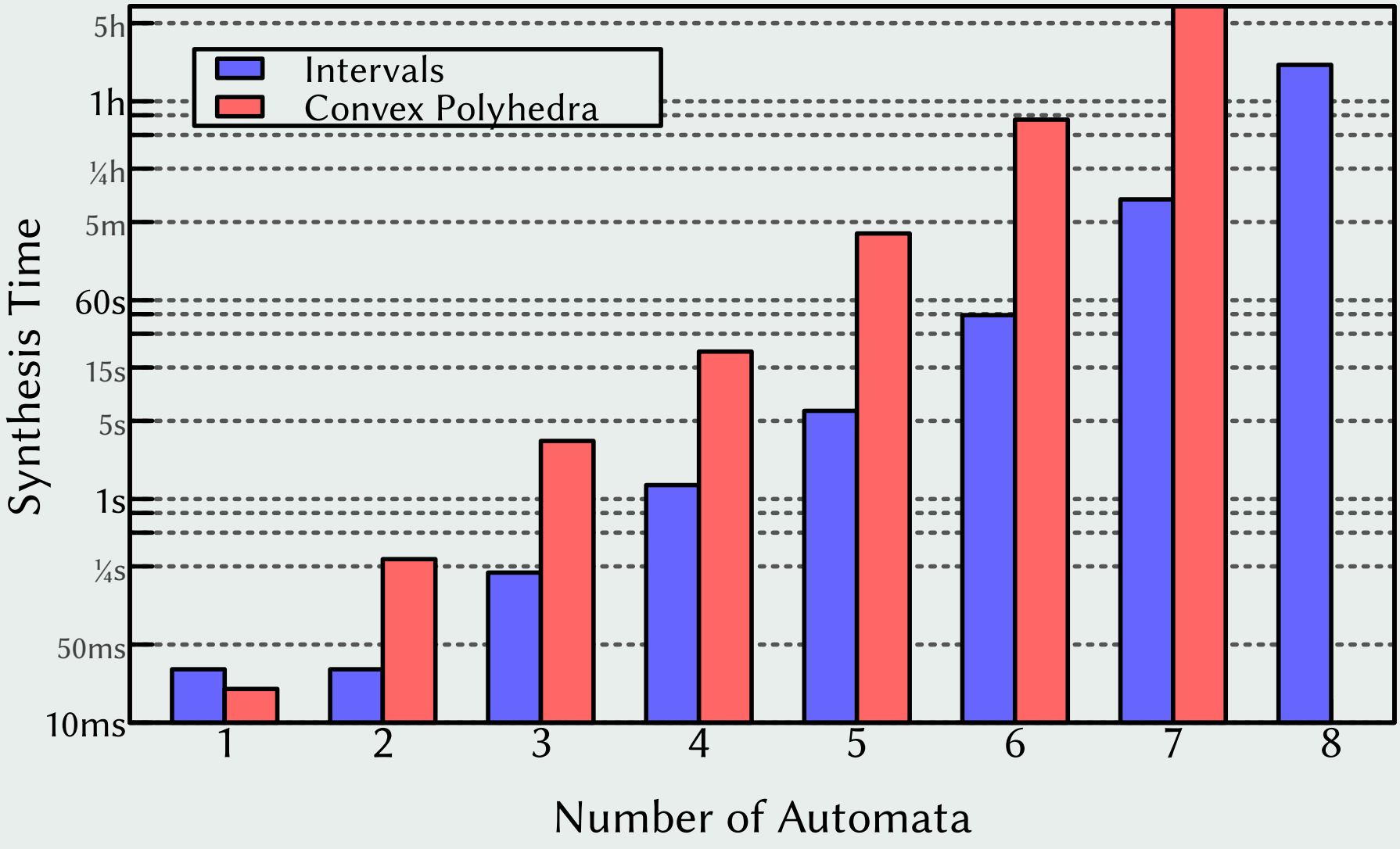
Performance Evaluation (cont'd)

Results for Mutual Exclusion \ Bounds \ Relational Constraints}



Performance Evaluation (cont'd)

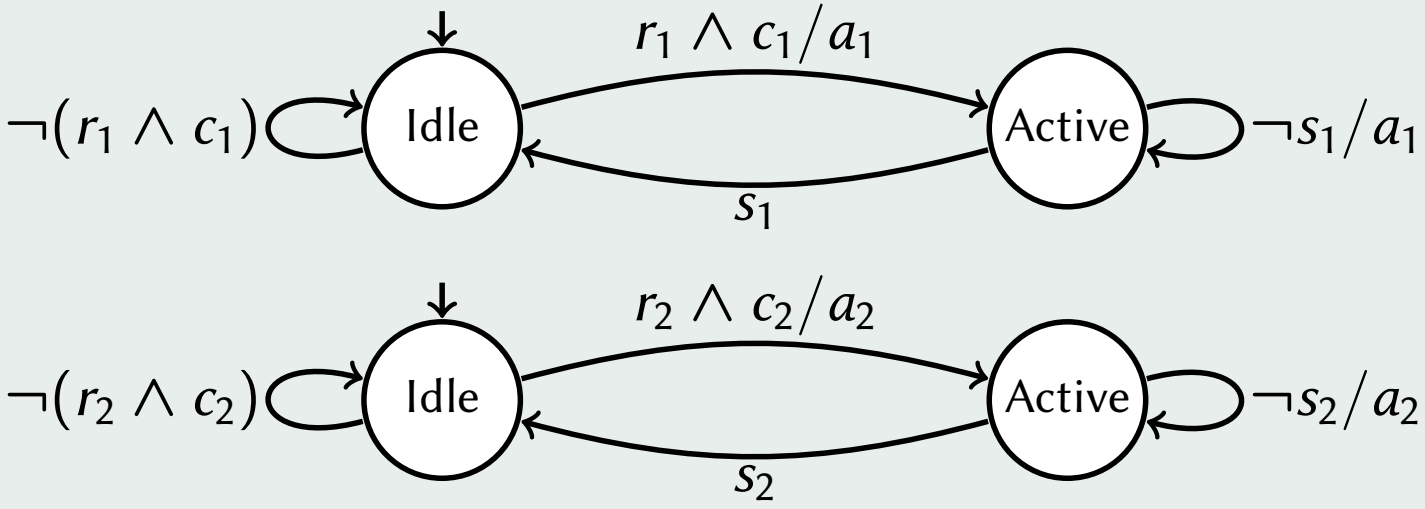
Results for Mutual Exclusion \ Bounds{ \ Relational Constraints}



Outline

Performance Comparison with SIGALI

Benchmark



- $X = \langle t_1, t_2, a_1, a_2 \rangle$ $\quad \mathcal{D}_X = \{\text{Idle}, \text{Active}\}^2 \times \mathbb{B}^2$
- $U = \langle r_1, r_2, s_1, s_2 \rangle, C = \langle c_1, c_2 \rangle$ $\quad \mathcal{D}_{l_{uc}} = \mathbb{B}^4, \mathcal{D}_{l_c} = \mathbb{B}^2$
- Enforcing Mutual Exclusion Between Active States

$$\Phi(\langle t_1, t_2, \ldots \rangle) = (t_1 \neq t_2 \vee t_1 = \text{Idle})$$

Performance Comparison with SIGALI (cont'd)

Comparing Execution Times