Accelerated Data-flow Analysis

Jérôme Leroux and Grégoire Sutre

LaBRI, Université de Bordeaux, CNRS, France

LSV Seminar, Laboratoire Spécification et Vérification, Cachan
May 27, 2008

Leroux & Sutre. Accelerated Data-flow Analysis. *SAS'07*.
Leroux & Sutre. Acceleration in Convex Data-Flow Analysis. *FSTTCS'07*.

Outline

Motivating Example

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if      (x ≥ 75)  x = x+5;
4    else if (x ≥ 50)  x = x−3;
5    else              x = x+2;
6  }
```

Question

Does the program leave the while loop ?

Answer

No iff the value of variable x at line 2 remains not greater than 100

Motivating Example

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if      (x ≥ 75)  x = x+5;
4    else if (x ≥ 50)  x = x−3;
5    else              x = x+2;
6  }
```

Question

Does the program leave the while loop ?

Answer

No iff the value of variable x at line 2 remains not greater than 100

Motivating Example (Reachability Set)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if       (x ≥ 75)   x = x+5;
4    else if  (x ≥ 50)   x = x−3;
5    else                x = x+2;
6  }
```

Reachability Set

Values of x at lines 1, 2, 3, 6:

$$
\begin{array}{lcl}
1 & \mapsto & \mathbb{Z} \\
2 & \mapsto & \{1,3,\ldots,51\} \cup \{48,50\} \\
3 & \mapsto & \{1,3,\ldots,51\} \cup \{48,50\} \\
6 & \mapsto & \{3,5,\ldots,51\} \cup \{48,50\}
\end{array}
$$

Features and Drawbacks

- The reachability set is the most precise invariant
- The computation of the reachability set may not terminate
- Its precision is often unnecessary to prove the property of interest

Motivating Example (Reachability Set)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if       (x ≥ 75)   x = x+5;
4    else if  (x ≥ 50)   x = x−3;
5    else                x = x+2;
6  }
```

Reachability Set

Values of x at lines 1, 2, 3, 6:

$$
\begin{array}{lcl}
1 & \mapsto & \mathbb{Z} \\
2 & \mapsto & \{1, 3, \ldots, 51\} \cup \{48, 50\} \\
3 & \mapsto & \{1, 3, \ldots, 51\} \cup \{48, 50\} \\
6 & \mapsto & \{3, 5, \ldots, 51\} \cup \{48, 50\}
\end{array}
$$

Features and Drawbacks

- The reachability set is the most precise invariant
- The computation of the reachability set may not terminate
- Its precision is often unnecessary to prove the property of interest

Motivating Example (Interval Analysis with ∇ and Δ)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if      (x ≥ 75)   x = x+5;
4    else if (x ≥ 50)   x = x−3;
5    else               x = x+2;
6  }
```

Interval Analysis with ∇ & Δ

Bounds of x at lines 1, 2, 3, 6:

Features and Drawbacks

- The use of widening (∇) ensures termination of the analysis
- The use of narrowing (Δ) improves precision
- The invariant may be too coarse to prove the property of interest

Motivating Example (Interval Analysis with ∇ and Δ)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if       (x ≥ 75)   x = x+5;
4    else if  (x ≥ 50)   x = x−3;
5    else                x = x+2;
6  }
```

Interval Analysis with ∇ & Δ

Bounds of x at lines 1, 2, 3, 6:

$$1 \quad \mapsto \quad]-\infty, +\infty[$$

Features and Drawbacks

- The use of widening (∇) ensures termination of the analysis
- The use of narrowing (Δ) improves precision
- The invariant may be too coarse to prove the property of interest

Motivating Example (Interval Analysis with ∇ and Δ)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if      (x ≥ 75)   x = x+5;
4    else if (x ≥ 50)   x = x−3;
5    else               x = x+2;
6  }
```

Interval Analysis with ∇ & Δ

Bounds of x at lines 1, 2, 3, 6:

$$1 \mapsto]-\infty, +\infty[$$
$$2 \mapsto \{1\}$$

Features and Drawbacks

- The use of widening (∇) ensures termination of the analysis
- The use of narrowing (Δ) improves precision
- The invariant may be too coarse to prove the property of interest

Motivating Example (Interval Analysis with ∇ and Δ)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if      (x ≥ 75)   x = x+5;
4    else if (x ≥ 50)   x = x−3;
5    else               x = x+2;
6  }
```

Interval Analysis with ∇ & Δ

Bounds of x at lines 1, 2, 3, 6:

$$\begin{array}{lcl} 1 & \mapsto &]-\infty,+\infty[\\ 2 & \mapsto & \{1\} \\ 3 & \mapsto & \{1\} \end{array}$$

Features and Drawbacks

- The use of widening (∇) ensures termination of the analysis
- The use of narrowing (Δ) improves precision
- The invariant may be too coarse to prove the property of interest

Motivating Example (Interval Analysis with ∇ and Δ)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if      (x ≥ 75)   x = x+5;
4    else if (x ≥ 50)   x = x−3;
5    else               x = x+2;
6  }
```

Interval Analysis with ∇ & Δ

Bounds of x at lines 1, 2, 3, 6:

$$\begin{array}{lcl} 1 & \mapsto &]-\infty,+\infty[\\ 2 & \mapsto & \{1\} \\ 3 & \mapsto & \{1\} \\ 6 & \mapsto & \{3\} \end{array}$$

Features and Drawbacks

- The use of widening (∇) ensures termination of the analysis
- The use of narrowing (Δ) improves precision
- The invariant may be too coarse to prove the property of interest

Motivating Example (Interval Analysis with ∇ and Δ)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if       (x ≥ 75)   x = x+5;
4    else if  (x ≥ 50)   x = x−3;
5    else                x = x+2;
6  }
```

Interval Analysis with ∇ & Δ

Bounds of x at lines 1, 2, 3, 6:

$$
\begin{array}{lcl}
1 & \mapsto &]-\infty,+\infty[\\
2 & \mapsto & \{1\} \sqcup \{3\} = [1,3] \\
3 & \mapsto & \{1\} \\
6 & \mapsto & \{3\}
\end{array}
$$

Features and Drawbacks

- The use of widening (∇) ensures termination of the analysis
- The use of narrowing (Δ) improves precision
- The invariant may be too coarse to prove the property of interest

Motivating Example (Interval Analysis with ∇ and Δ)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if       (x ≥ 75)   x = x+5;
4    else if  (x ≥ 50)   x = x−3;
5    else                x = x+2;
6  }
```

Interval Analysis with ∇ & Δ

Bounds of x at lines 1, 2, 3, 6:

$$
\begin{array}{lcl}
1 & \mapsto &]-\infty, +\infty[\\
2 & \mapsto & [1, 3] \\
3 & \mapsto & \{1\} \\
6 & \mapsto & \{3\}
\end{array}
$$

Features and Drawbacks

- The use of widening (∇) ensures termination of the analysis
- The use of narrowing (Δ) improves precision
- The invariant may be too coarse to prove the property of interest

Motivating Example (Interval Analysis with ∇ and Δ)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if      (x ≥ 75)   x = x+5;
4    else if (x ≥ 50)   x = x−3;
5    else               x = x+2;
6  }
```

Interval Analysis with ∇ & Δ

Bounds of x at lines 1, 2, 3, 6:

$$
\begin{array}{lcl}
1 & \mapsto &]-\infty, +\infty[\\
2 & \mapsto & [1, 3] \\
3 & \mapsto & [1, 3] \\
6 & \mapsto & \{3\}
\end{array}
$$

Features and Drawbacks

- The use of widening (∇) ensures termination of the analysis
- The use of narrowing (Δ) improves precision
- The invariant may be too coarse to prove the property of interest

Motivating Example (Interval Analysis with ∇ and Δ)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if       (x ≥ 75)   x = x+5;
4    else if  (x ≥ 50)   x = x−3;
5    else                x = x+2;
6  }
```

Interval Analysis with ∇ & Δ

Bounds of x at lines 1, 2, 3, 6:

$$\begin{array}{lcl} 1 & \mapsto &]-\infty,+\infty[\\ 2 & \mapsto & [1,3] \\ 3 & \mapsto & [1,3] \\ 6 & \mapsto & [3,5] \end{array}$$

Features and Drawbacks

- The use of widening (∇) ensures termination of the analysis
- The use of narrowing (Δ) improves precision
- The invariant may be too coarse to prove the property of interest

Motivating Example (Interval Analysis with ∇ and Δ)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if      (x ≥ 75)   x = x+5;
4    else if (x ≥ 50)   x = x−3;
5    else               x = x+2;
6  }
```

Interval Analysis with ∇ & Δ

Bounds of x at lines 1, 2, 3, 6:

$$
\begin{array}{lcl}
1 & \mapsto &]-\infty,+\infty[\\
2 & \mapsto & [1,3] \nabla [3,5] \\
3 & \mapsto & [1,3] \\
6 & \mapsto & [3,5]
\end{array}
$$

Features and Drawbacks

- The use of widening (∇) ensures termination of the analysis
- The use of narrowing (Δ) improves precision
- The invariant may be too coarse to prove the property of interest

Motivating Example (Interval Analysis with ∇ and Δ)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if      (x ≥ 75)   x = x+5;
4    else if (x ≥ 50)   x = x−3;
5    else               x = x+2;
6  }
```

Interval Analysis with ∇ & Δ

Bounds of x at lines 1, 2, 3, 6:

$$
\begin{array}{lcl}
1 & \mapsto &]-\infty, +\infty[\\
2 & \mapsto & [1, +\infty[\\
3 & \mapsto & [1, 3] \\
6 & \mapsto & [3, 5]
\end{array}
$$

Features and Drawbacks

- The use of widening (∇) ensures termination of the analysis
- The use of narrowing (Δ) improves precision
- The invariant may be too coarse to prove the property of interest

Motivating Example (Interval Analysis with ∇ and Δ)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if      (x ≥ 75)   x = x+5;
4    else if (x ≥ 50)   x = x−3;
5    else               x = x+2;
6  }
```

Interval Analysis with ∇ & Δ

Bounds of x at lines 1, 2, 3, 6:

$$
\begin{array}{lcl}
1 & \mapsto &]-\infty, +\infty[\\
2 & \mapsto & [1, +\infty[\\
3 & \mapsto & [1, 100] \\
6 & \mapsto & [3, 5]
\end{array}
$$

Features and Drawbacks

- The use of widening (∇) ensures termination of the analysis
- The use of narrowing (Δ) improves precision
- The invariant may be too coarse to prove the property of interest

Motivating Example (Interval Analysis with ∇ and Δ)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if       (x ≥ 75)   x = x+5;
4    else if  (x ≥ 50)   x = x−3;
5    else                x = x+2;
6  }
```

Interval Analysis with ∇ & Δ

Bounds of x at lines 1, 2, 3, 6:

$$
\begin{array}{lcl}
1 & \mapsto &]-\infty, +\infty[\\
2 & \mapsto & [1, +\infty[\\
3 & \mapsto & [1, 100] \\
6 & \mapsto & [3, 105]
\end{array}
$$

Features and Drawbacks

- The use of widening (∇) ensures termination of the analysis
- The use of narrowing (Δ) improves precision
- The invariant may be too coarse to prove the property of interest

Motivating Example (Interval Analysis with ∇ and Δ)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if       (x ≥ 75)   x = x+5;
4    else if  (x ≥ 50)   x = x−3;
5    else                x = x+2;
6  }
```

Interval Analysis with ∇ & Δ

Bounds of x at lines 1, 2, 3, 6:

$$
\begin{array}{lcl}
1 & \mapsto &]-\infty,+\infty[\\
2 & \mapsto & [1,+\infty[\,\Delta\,[3,105] \\
3 & \mapsto & [1,100] \\
6 & \mapsto & [3,105]
\end{array}
$$

Features and Drawbacks

- The use of widening (∇) ensures termination of the analysis
- The use of narrowing (Δ) improves precision
- The invariant may be too coarse to prove the property of interest

Motivating Example (Interval Analysis with ∇ and Δ)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if      (x ≥ 75)   x = x+5;
4    else if (x ≥ 50)   x = x−3;
5    else               x = x+2;
6  }
```

Interval Analysis with ∇ & Δ

Bounds of x at lines 1, 2, 3, 6:

$$
\begin{array}{lcl}
1 & \mapsto &]-\infty,+\infty[\\
2 & \mapsto & [1,105] \\
3 & \mapsto & [1,100] \\
6 & \mapsto & [3,105]
\end{array}
$$

Features and Drawbacks

- The use of widening (∇) ensures termination of the analysis
- The use of narrowing (Δ) improves precision
- The invariant may be too coarse to prove the property of interest

Motivating Example (Interval Analysis with ∇ and Δ)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if      (x ≥ 75)   x = x+5;
4    else if (x ≥ 50)   x = x−3;
5    else               x = x+2;
6  }
```

Interval Analysis with ∇ & Δ

Bounds of x at lines 1, 2, 3, 6:

$$
\begin{array}{lcl}
1 & \mapsto &]-\infty, +\infty[\\
2 & \mapsto & [1, 105] \\
3 & \mapsto & [1, 100] \\
6 & \mapsto & [3, 105]
\end{array}
$$

Features and Drawbacks

- The use of widening (∇) ensures termination of the analysis
- The use of narrowing (Δ) improves precision
- The invariant may be too coarse to prove the property of interest

Motivating Example (Exact Interval Analysis)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if       (x ≥ 75)   x = x+5;
4    else if  (x ≥ 50)   x = x−3;
5    else                x = x+2;
6  }
```

Exact Interval Analysis

Bounds of x at lines 1, 2, 3, 6:

$$
\begin{array}{lcl}
1 & \mapsto &]-\infty, +\infty[\\
2 & \mapsto & [1, 51] \\
3 & \mapsto & [1, 51] \\
6 & \mapsto & [3, 51]
\end{array}
$$

Observations

- Intervals are actually sufficient to prove the property of interest, i.e. that *this* program never leaves the while loop
- Imprecision in the previous analysis came from widening

Motivating Example (Exact Interval Analysis)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if       (x ≥ 75)   x = x+5;
4    else if  (x ≥ 50)   x = x−3;
5    else                x = x+2;
6  }
```

Exact Interval Analysis

Bounds of x at lines 1, 2, 3, 6:

$$\begin{array}{lcl} 1 & \mapsto &]-\infty,+\infty[\\ 2 & \mapsto & [1,51] \\ 3 & \mapsto & [1,51] \\ 6 & \mapsto & [3,51] \end{array}$$

Observations

- Intervals are actually sufficient to prove the property of interest, i.e. that *this* program never leaves the while loop
- Imprecision in the previous analysis came from widening

Problematics

Invariants for Verification of Safety Properties

- Efficient computation of precise enough invariants
- Data-flow analysis, abstract interpretation
- Widenings / narrowings: successful approach, but might lead to invariants to coarse for verification

Our Objective

Computation of the exact solution to data-flow analysis problems

- Meet Over all Paths
- Minimum Fix Point

Challenge

- When and how can exact analysis be algorithmically performed ?

Problematics

Invariants for Verification of Safety Properties

- Efficient computation of precise enough invariants
- Data-flow analysis, abstract interpretation
- Widenings / narrowings: successful approach, but might lead to invariants to coarse for verification

Our Objective

Computation of the exact solution to data-flow analysis problems

- Meet Over all Paths
- Minimum Fix Point

Challenge

- When and how can exact analysis be algorithmically performed ?

Problematics

Invariants for Verification of Safety Properties

- Efficient computation of precise enough invariants
- Data-flow analysis, abstract interpretation
- Widenings / narrowings: successful approach, but might lead to invariants to coarse for verification

Our Objective

Computation of the exact solution to data-flow analysis problems

- Meet Over all Paths
- Minimum Fix Point

Challenge

- When and how can exact analysis be algorithmically performed ?

Our Approach

Acceleration in Symbolic Verification

- Symbolically compute the effect of iterating a given cycle
- Speed up Kleene fix-point iteration in concrete data-flow analysis
- Developped for several data types: integer variables, continuous variables, fifo queues, ...
- Implemented in tools (LASH, FAST, TREX)
- No theoretical termination guarantee, but good results in practice

This Work

- Extend acceleration to abstract data-flow analysis
- Apply the framework to convex / intervals data-flow analysis
- Investigate completeness of the approach

Our Approach

Acceleration in Symbolic Verification

- Symbolically compute the effect of iterating a given cycle
- Speed up Kleene fix-point iteration in concrete data-flow analysis
- Developped for several data types: integer variables, continuous variables, fifo queues, ...
- Implemented in tools (LASH, FAST, TREX)
- No theoretical termination guarantee, but good results in practice

This Work

- Extend acceleration to abstract data-flow analysis
- Apply the framework to convex / intervals data-flow analysis
- Investigate completeness of the approach

Outline

Outline

Data-Flow Programs (Syntax)

Consider a complete lattice $(A, \sqsubseteq)$ and a finite set $\mathcal{X}$ of *variables*

Definition

A transition on $\mathcal{X}$ is any tuple $\langle X_1, \ldots, X_n; f; X \rangle$ where :

- $X_1, \ldots, X_n \in \mathcal{X}$ are pairwise disjoint *input variables*
- $f \in A^n \to A$ is a monotonic *transfer function*
- $X \in \mathcal{X}$ is an *output variable*

Notation : $\langle X_1, \ldots, X_n; f; X \rangle$ is also written $X := f(X_1, \ldots, X_n)$

Definition

A data-flow program over $(A, \sqsubseteq)$ is any pair $\mathcal{S} = (\mathcal{X}, T)$ where:

- $\mathcal{X}$ is a finite set of *variables*
- T is a finite set of *transitions* on $\mathcal{X}$

Data-Flow Programs (Syntax)

Consider a complete lattice $(A, \sqsubseteq)$ and a finite set $\mathcal{X}$ of *variables*

Definition

A transition on $\mathcal{X}$ is any tuple $\langle X_1, \ldots, X_n; f; X \rangle$ where :

- $X_1, \ldots, X_n \in \mathcal{X}$ are pairwise disjoint *input variables*
- $f \in A^n \to A$ is a monotonic *transfer function*
- $X \in \mathcal{X}$ is an *output variable*

Notation : $\langle X_1, \ldots, X_n; f; X \rangle$ is also written $X := f(X_1, \ldots, X_n)$

Definition

A data-flow program over $(A, \sqsubseteq)$ is any pair $\mathcal{S} = (\mathcal{X}, T)$ where:

- $\mathcal{X}$ is a finite set of *variables*
- T is a finite set of *transitions* on $\mathcal{X}$

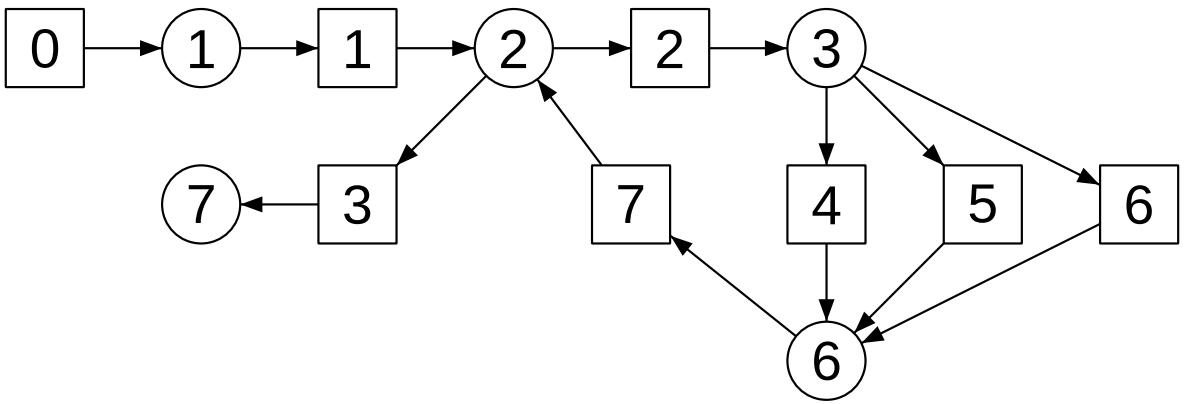
Motivating Example (Intervals)

Example (Code Snippet)

```
1  x = 1;
2  while (x ≤ 100) {
3    if       (x ≥ 75)   x = x+5;
4    else if  (x ≥ 50)   x = x−3;
5    else                x = x+2;
6  }
7
```

Data-Flow Program

$$
\begin{array}{ll}
(t_0) & X_1 := \top \\
(t_1) & X_2 := (\{0\} \,.\, X_1) + \{1\} \\
(t_2) & X_3 := X_2 \sqcap]-\infty, 100] \\
(t_3) & X_7 := X_2 \sqcap [101, +\infty[\\
(t_4) & X_6 := (X_3 \sqcap [75, +\infty[) + \{5\} \\
(t_5) & X_6 := (X_3 \sqcap [50, 74]) - \{3\} \\
(t_6) & X_6 := (X_3 \sqcap]-\infty, 49]) + \{2\} \\
(t_7) & X_2 := X_6
\end{array}
$$



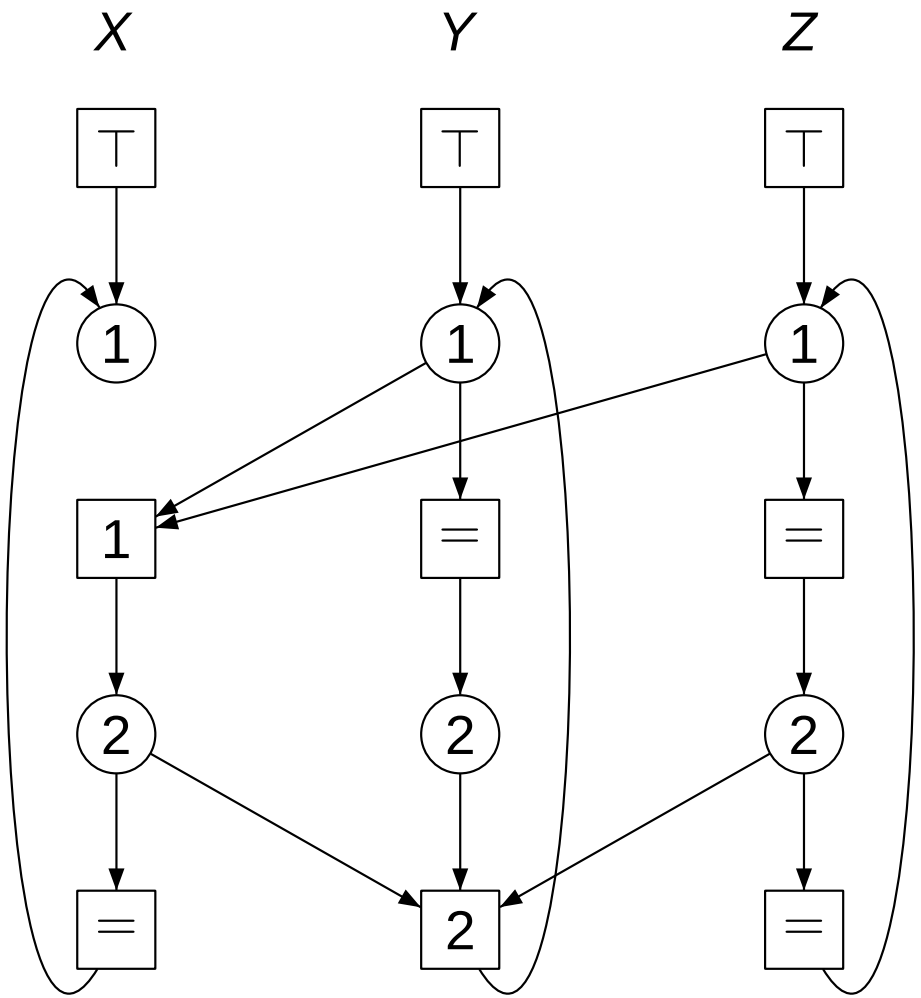
Example with Multiple Inputs (Intervals)

Example

```
1  for (;;) { x = y + z;
2             y = x*z − y; }
3
```

Data-Flow Program

$$
\begin{array}{ll}
 & X_1 := \top \\
 & Y_1 := \top \\
 & Z_1 := \top \\
(t_1) & X_2 := Y_1 + Z_1 \\
(t_2) & Y_1 := (X_2 * Z_2) - Y_2 \\
 & X_3 := X_1 \sqcap \bot \\
 & Y_3 := Y_1 \sqcap \bot \\
 & Z_3 := Z_1 \sqcap \bot
\end{array}
$$



Self-Loop GTS Example (2-Dim Closed Convex)

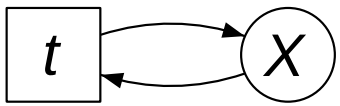
Example

```
1  while (x ≥ 0 ∧ y ≥ 0) {  x = x−1; y = y+1; }
```

Data-Flow Program

- Lattice: closed convex subsets of $\mathbb{R}^2$
- $\mathcal{X} = \{X\}$
- $T = \{t\}$
- $(t) \quad X := (G \cap X) + \vec{d}$

$$\text{with} \begin{cases} G & = & \mathbb{R}^2_+ \\ \vec{d} & = & (-1, 1) \end{cases}$$



Self-Loop GTS Example (2-Dim Closed Convex)

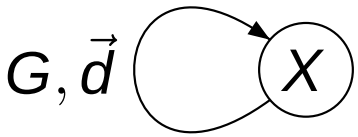
Example

```
1  while (x ≥ 0 ∧ y ≥ 0) { x = x−1; y = y+1; }
```

Data-Flow Program

- Lattice: closed convex subsets of $\mathbb{R}^2$
- $\mathcal{X} = \{X\}$
- $T = \{t\}$
- $(t) \quad X := (G \cap X) + \vec{d}$

with $\begin{cases} G & = & \mathbb{R}^2_+ \\ \vec{d} & = & (-1, 1) \end{cases}$



Data-Flow Programs (Semantics)

Definition (Recall)

A data-flow program over $(A, \sqsubseteq)$ is any pair $\mathcal{S} = (\mathcal{X}, T)$ where:

- $\mathcal{X}$ is a finite set of *variables*
- T is a finite set of *transitions* on $\mathcal{X}$

$(A, \sqsubseteq)$ is extended to the complete lattice of valuations $(\mathcal{X} \to A, \sqsubseteq)$

Definition

The data-flow semantics $[\![t]\!]$ of any transition $t = X := f(X_1, \ldots, X_n)$ is the monotonic function in $(\mathcal{X} \to A) \to (\mathcal{X} \to A)$ defined by:

$$\begin{aligned} [\![t]\!](\rho)(X) &= f(\rho(X_1), \ldots, \rho(X_n)) \\ [\![t]\!](\rho)(Y) &= \rho(Y) \text{ for all } Y \neq X \end{aligned}$$

Initialized Data-Flow Programs

Definition

An initialized data-flow program over $(A, \sqsubseteq)$ is any pair $(\mathcal{S}, \rho_0)$ where:

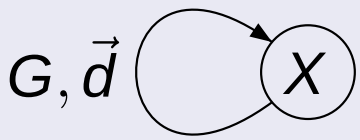
- $\mathcal{S} = (\mathcal{X}, T)$ is a data-flow program over $(A, \sqsubseteq)$
- $\rho_0 : \mathcal{X} \to A$ is an *initial valuation*

We identify $(\mathcal{S}, \rho_0)$ with the data-flow program:

$$\mathcal{S}' = (\mathcal{X}, T \cup \{X := \rho_0(X) \mid X \in \mathcal{X}\})$$

Self-Loop GTS Example (2-Dim Closed Convex)

Data-Flow Program

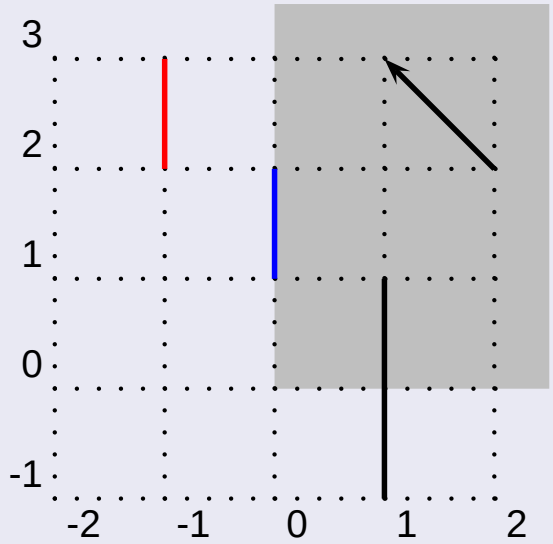


- $\mathcal{X} = \{X\}$
- $T = \{t\}$
- $(t) \quad X := (G \cap X) + \vec{d}$

  with $\begin{cases} G & = & \mathbb{R}^2_+ \\ \vec{d} & = & (-1, 1) \end{cases}$

- $\rho_0 = \{X \mapsto 1 \times [-1, 1]\}$

Semantics



$$\begin{array}{lcl} \rho_1 & = & [\![t]\!](\rho_0) \\ \rho_2 & = & [\![t]\!](\rho_1) \\ \rho_3 & = & [\![t]\!](\rho_2) = \{X \mapsto \emptyset\} \end{array}$$

Data-Flow Analysis MFP et MOP Solutions

Minimum Fix-Point (MFP) Solution

$$\text{MFP}(\mathcal{S}) \quad = \quad \bigsqcap \{\rho \in \mathcal{X} \to A \mid [\![t]\!](\rho) \sqsubseteq \rho \text{ for all } t \in T\}$$

- MFP($\mathcal{S}$) is the least fix-point of $[\![T]\!] = \bigsqcup_{t \in T} [\![t]\!]$

Meet Over all Paths (MOP) Solution

$$\text{MOP}(\mathcal{S}) \quad = \quad \bigsqcup \{[\![t_k]\!] \circ \cdots \circ [\![t_1]\!](\bot) \mid t_1 \cdots t_k \in T^*\}$$

- Can be viewed as the “abstraction” of the reachability set

Kleene Fix-Point Iteration

$$\text{MOP}(\mathcal{S}) \quad \sqsubseteq \quad \bigsqcup_{i \in \mathbb{N}} [\![T]\!]^i(\bot) \quad \sqsubseteq \quad \text{MFP}(\mathcal{S})$$

Data-Flow Analysis MFP et MOP Solutions

Minimum Fix-Point (MFP) Solution

$$\text{MFP}(\mathcal{S}) \quad = \quad \bigsqcap \{\rho \in \mathcal{X} \to A \mid [\![t]\!](\rho) \sqsubseteq \rho \text{ for all } t \in T\}$$

- $\text{MFP}(\mathcal{S})$ is the least fix-point of $[\![T]\!] = \bigsqcup_{t \in T} [\![t]\!]$

Meet Over all Paths (MOP) Solution

$$\text{MOP}(\mathcal{S}) \quad = \quad \bigsqcup \{[\![t_k]\!] \circ \cdots \circ [\![t_1]\!](\bot) \mid t_1 \cdots t_k \in T^*\}$$

- Can be viewed as the “abstraction” of the reachability set

Kleene Fix-Point Iteration

$$\text{MOP}(\mathcal{S}) \quad \sqsubseteq \quad \bigsqcup_{i \in \mathbb{N}} [\![T]\!]^i(\bot) \quad \sqsubseteq \quad \text{MFP}(\mathcal{S})$$

Data-Flow Analysis MFP et MOP Solutions

Minimum Fix-Point (MFP) Solution

$$\text{MFP}(\mathcal{S}) \quad = \quad \bigsqcap \{\rho \in \mathcal{X} \to A \mid [\![t]\!](\rho) \sqsubseteq \rho \text{ for all } t \in T\}$$

- $\text{MFP}(\mathcal{S})$ is the least fix-point of $[\![T]\!] = \bigsqcup_{t \in T} [\![t]\!]$

Meet Over all Paths (MOP) Solution

$$\text{MOP}(\mathcal{S}) \quad = \quad \bigsqcup \{[\![t_k]\!] \circ \cdots \circ [\![t_1]\!](\bot) \mid t_1 \cdots t_k \in T^*\}$$

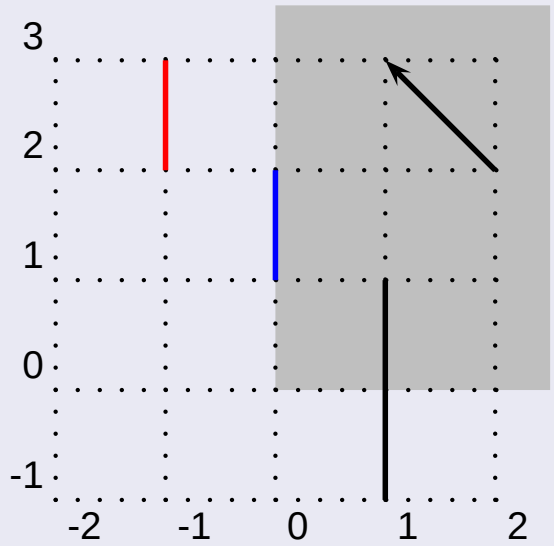
- Can be viewed as the “abstraction” of the reachability set

Kleene Fix-Point Iteration

$$\text{MOP}(\mathcal{S}) \quad \sqsubseteq \quad \bigsqcup_{i \in \mathbb{N}} [\![T]\!]^i(\bot) \quad \sqsubseteq \quad \text{MFP}(\mathcal{S})$$

Self-Loop GTS Example (2-Dim Closed Convex)

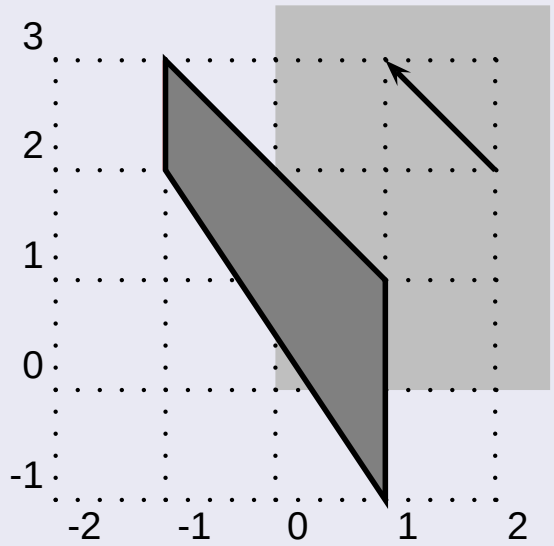
MOP Solution



$$\begin{aligned} \rho_1 &= [\![t]\!](\rho_0) \\ \rho_2 &= [\![t]\!](\rho_1) \\ \rho_3 &= [\![t]\!](\rho_2) = \{X \mapsto \emptyset\} \end{aligned}$$

Self-Loop GTS Example (2-Dim Closed Convex)

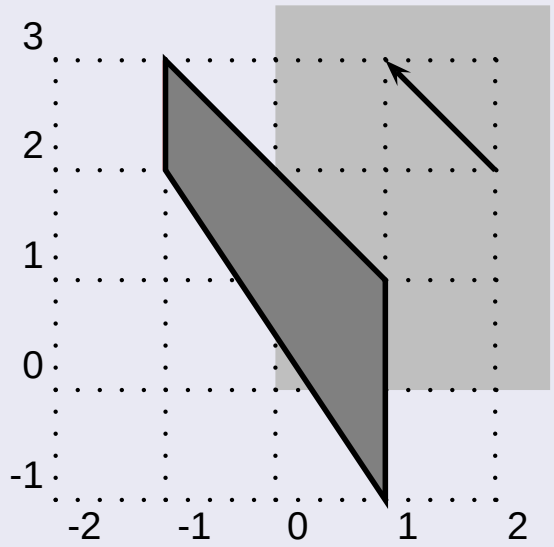
MOP Solution



$$\begin{aligned} \rho_1 &= [\![t]\!](\rho_0) \\ \rho_2 &= [\![t]\!](\rho_1) \\ \rho_3 &= [\![t]\!](\rho_2) = \{X \mapsto \emptyset\} \end{aligned}$$

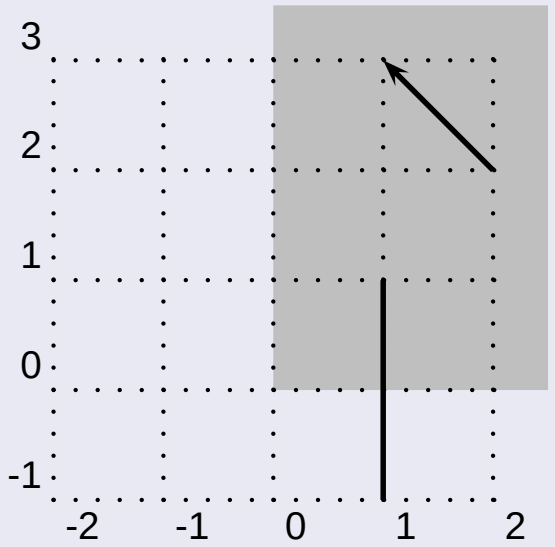
Self-Loop GTS Example (2-Dim Closed Convex)

MOP Solution



$$\begin{aligned} \rho_1 &= [\![t]\!](\rho_0) \\ \rho_2 &= [\![t]\!](\rho_1) \\ \rho_3 &= [\![t]\!](\rho_2) = \{X \mapsto \emptyset\} \end{aligned}$$

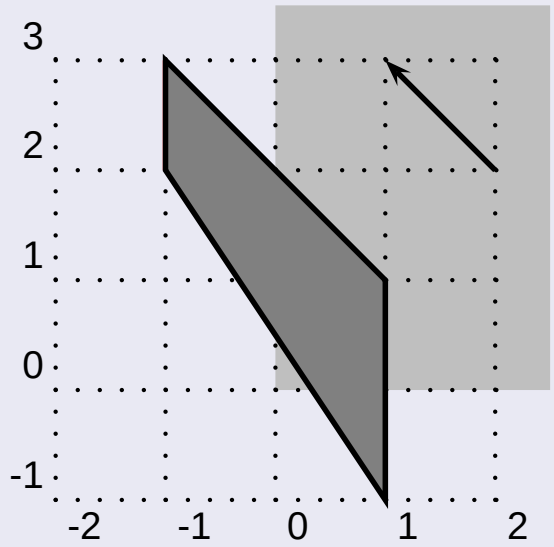
MFP Solution



$$\tau^1(\bot) = \rho_0$$

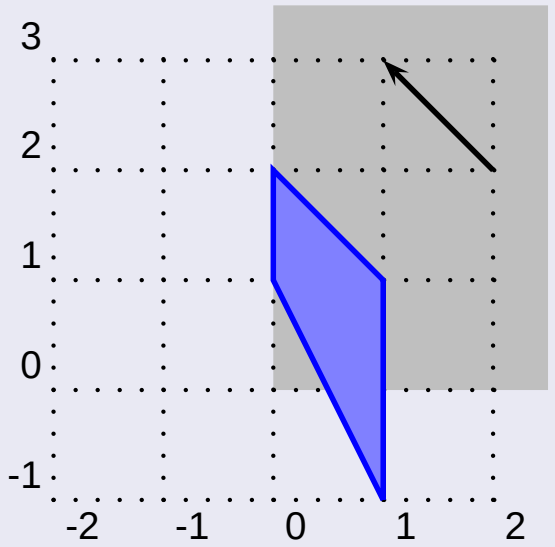
Self-Loop GTS Example (2-Dim Closed Convex)

MOP Solution



$$\begin{aligned} \rho_1 &= [\![t]\!](\rho_0) \\ \rho_2 &= [\![t]\!](\rho_1) \\ \rho_3 &= [\![t]\!](\rho_2) = \{X \mapsto \emptyset\} \end{aligned}$$

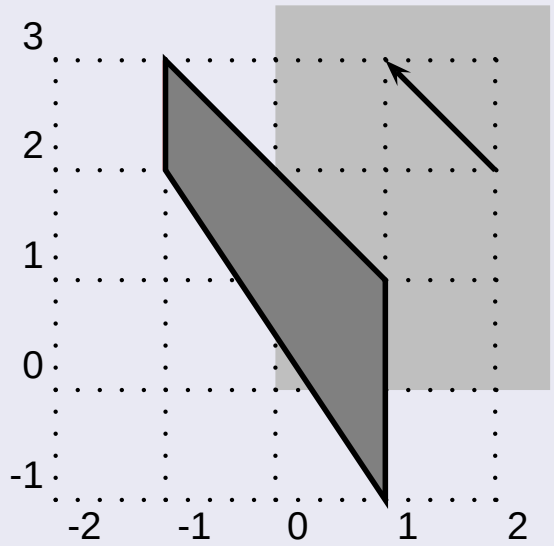
MFP Solution



$$\begin{aligned} \tau^1(\bot) &= \rho_0 \\ \tau^2(\bot) &= \rho_0 \sqcup [\![t]\!](\tau^1(\bot)) \end{aligned}$$

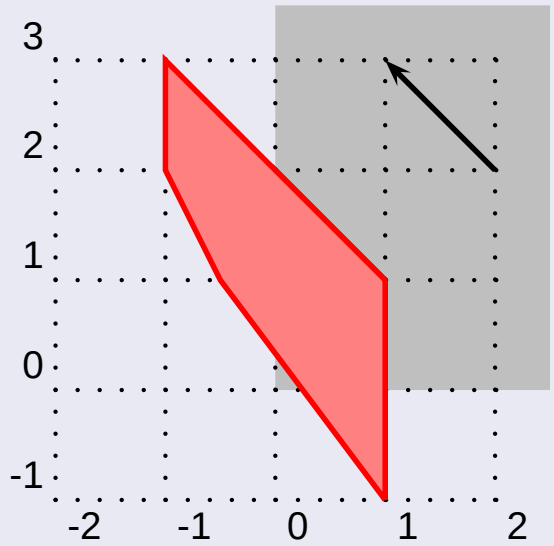
Self-Loop GTS Example (2-Dim Closed Convex)

MOP Solution



$$\begin{aligned} \rho_1 &= [\![t]\!](\rho_0) \\ \rho_2 &= [\![t]\!](\rho_1) \\ \rho_3 &= [\![t]\!](\rho_2) = \{X \mapsto \emptyset\} \end{aligned}$$

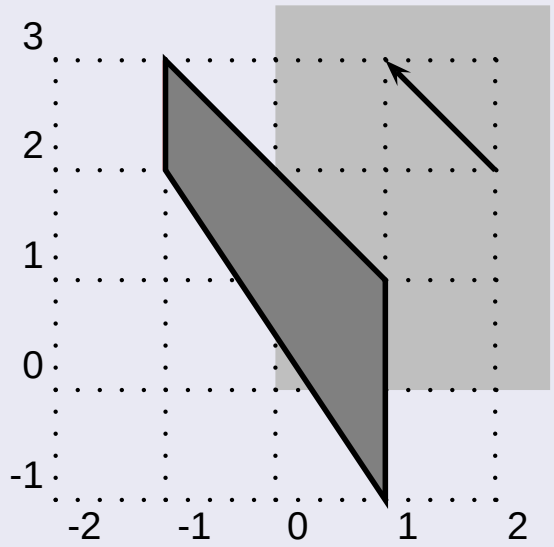
MFP Solution



$$\begin{aligned} \tau^1(\bot) &= \rho_0 \\ \tau^2(\bot) &= \rho_0 \sqcup [\![t]\!](\tau^1(\bot)) \\ \tau^i(\bot) &= \rho_0 \sqcup [\![t]\!](\tau^{i-1}(\bot)) \end{aligned}$$

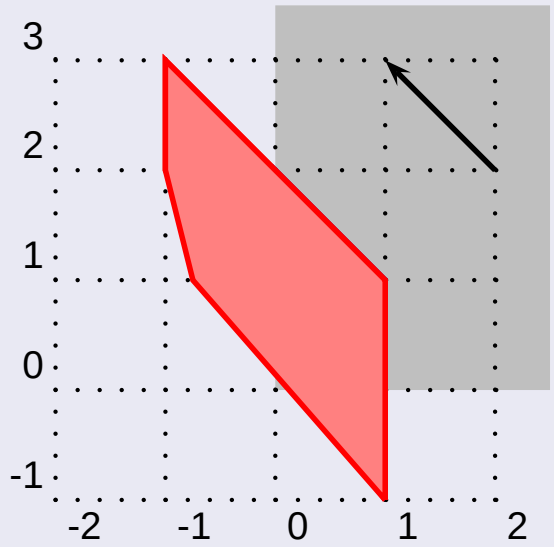
Self-Loop GTS Example (2-Dim Closed Convex)

MOP Solution



$$\begin{aligned} \rho_1 &= [\![t]\!](\rho_0) \\ \rho_2 &= [\![t]\!](\rho_1) \\ \rho_3 &= [\![t]\!](\rho_2) = \{X \mapsto \emptyset\} \end{aligned}$$

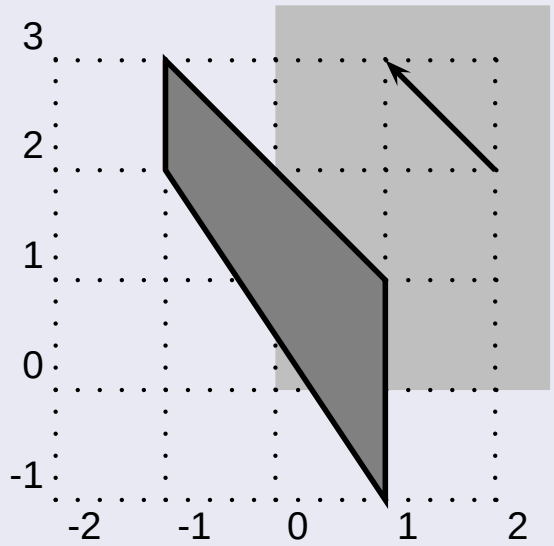
MFP Solution



$$\begin{aligned} \tau^1(\bot) &= \rho_0 \\ \tau^2(\bot) &= \rho_0 \sqcup [\![t]\!](\tau^1(\bot)) \\ \tau^i(\bot) &= \rho_0 \sqcup [\![t]\!](\tau^{i-1}(\bot)) \end{aligned}$$

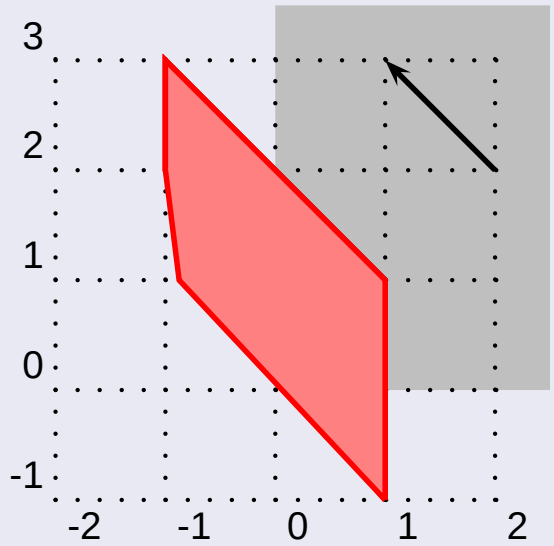
Self-Loop GTS Example (2-Dim Closed Convex)

MOP Solution



$$\begin{aligned} \rho_1 &= [\![t]\!](\rho_0) \\ \rho_2 &= [\![t]\!](\rho_1) \\ \rho_3 &= [\![t]\!](\rho_2) = \{X \mapsto \emptyset\} \end{aligned}$$

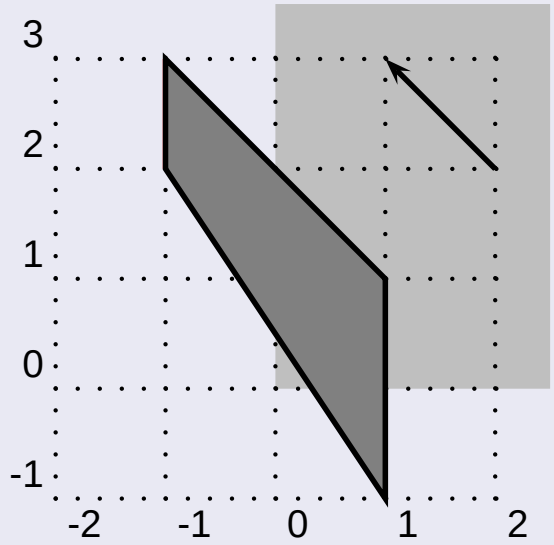
MFP Solution



$$\begin{aligned} \tau^1(\bot) &= \rho_0 \\ \tau^2(\bot) &= \rho_0 \sqcup [\![t]\!](\tau^1(\bot)) \\ \tau^i(\bot) &= \rho_0 \sqcup [\![t]\!](\tau^{i-1}(\bot)) \end{aligned}$$

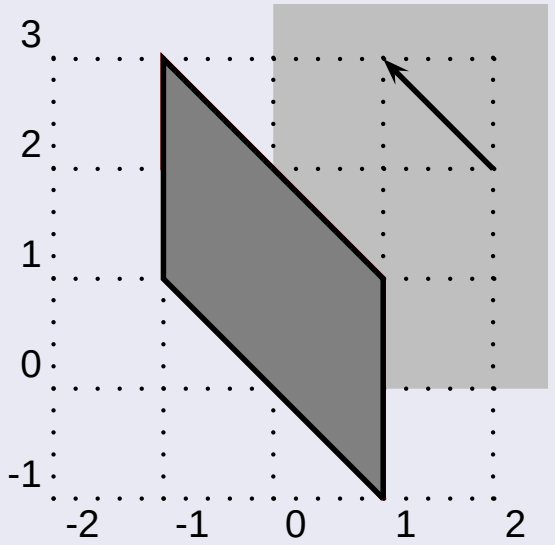
Self-Loop GTS Example (2-Dim Closed Convex)

MOP Solution



$$\begin{aligned} \rho_1 &= [\![t]\!](\rho_0) \\ \rho_2 &= [\![t]\!](\rho_1) \\ \rho_3 &= [\![t]\!](\rho_2) = \{X \mapsto \emptyset\} \end{aligned}$$

MFP Solution



Remark

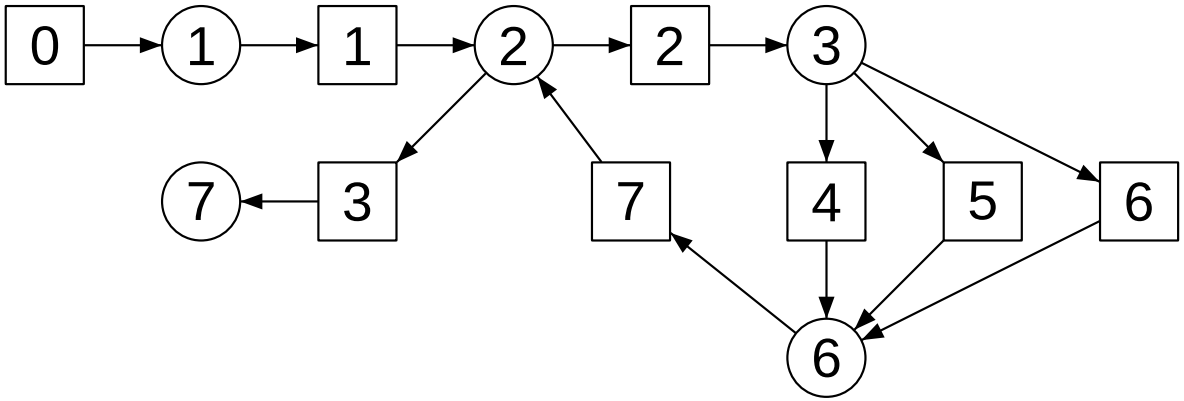
Kleene fix-point iteration does not stabilize

Acceleration of Cyclic Sub-Programs

Goal

Speed up Kleene fix-point iteration, without loosing precision

Idea : extract a cyclic sub-program and accelerate it!



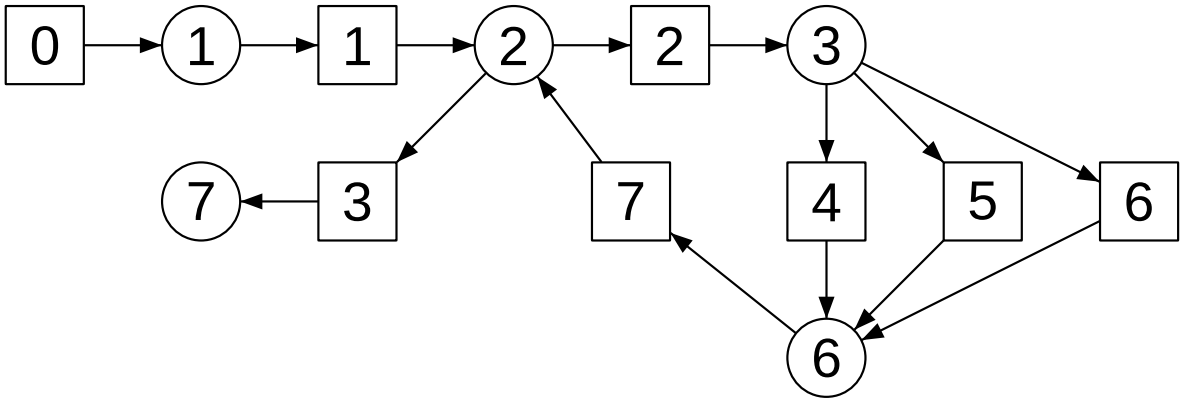
- Copies are allowed in the sub-program
- Renaming $\kappa : \mathcal{X}' \to \mathcal{X}$
- $\kappa^{-1}(\rho')(X) = \bigsqcup_{\kappa(X')=X} \rho'(X')$ (with $\kappa^{-1}(\rho')(X) = \bot$ if $X \notin \kappa(\mathcal{X}')$)

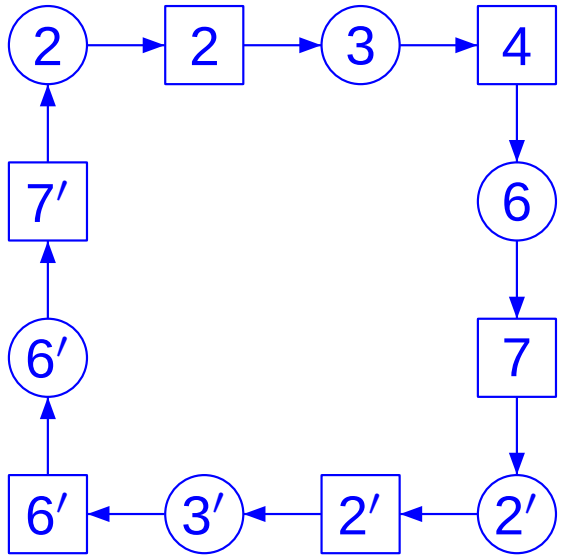
Acceleration of Cyclic Sub-Programs

Goal

Speed up Kleene fix-point iteration, without loosing precision

Idea : extract a cyclic sub-program and accelerate it!





- Copies are allowed in the sub-program
- Renaming $\kappa : \mathcal{X}' \to \mathcal{X}$
- $\kappa^{-1}(\rho')(X) = \bigsqcup_{\kappa(X')=X} \rho'(X')$ (with $\kappa^{-1}(\rho')(X) = \bot$ if $X \notin \kappa(\mathcal{X}')$)

Accelerated Computation of the MFP Solution (1)

1 AcceleratedMFP($\mathcal{S} = (\mathcal{X}, T)$: data-flow program over $(A, \sqsubseteq)$)
2 $\rho \leftarrow \bot$
3 `while` $[\![T]\!](\rho) \not\sqsubseteq \rho$ `do`
4 `if` $(*)$ `then`
5 select a transition $t \in T$
6 $\rho \leftarrow \rho \sqcup [\![t]\!](\rho)$
7 `else`
8 extract from $\mathcal{S}$ a cyclic sub-program $\mathcal{S}' = (\mathcal{X}', T')$
9 with renaming $\kappa \in \mathcal{X}' \to \mathcal{X}$
10 $\rho'_0 \leftarrow \rho \circ \kappa$
11 $\rho'' \leftarrow \text{MFP}(\mathcal{S}', \rho'_0)$ $\{\ \kappa^{-1}(\rho'') \sqsubseteq \text{MFP}(\mathcal{S}, \rho)\ \}$
12 $\rho \leftarrow \rho \sqcup \kappa^{-1}(\rho'')$
13 `return` ρ

Accelerated Computation of the MFP Solution (2)

10 $\rho'_0 \leftarrow \rho \circ \kappa$
11 $\rho'' \leftarrow \text{MFP}(\mathcal{S}', \rho'_0)$
12 $\rho \leftarrow \rho \sqcup \kappa^{-1}(\rho'')$

Correctness

$\kappa^{-1}(\rho'') \sqsubseteq \text{MFP}(\mathcal{S}, \rho)$

Alternatives

- line 10: any ρ'_0 such that $\rho'_0 \sqsubseteq \rho \circ \kappa$
  e.g. pick $X' \in \mathcal{X}'$ and define ρ'_0 by $\begin{cases} \rho'_0(X') & = & \rho \circ \kappa(X') \\ \rho'_0(Y') & = & \bot \text{ for all } Y' \neq X' \end{cases}$
- line 11: any ρ'' such that $\rho'' \sqsubseteq \text{MFP}(\mathcal{S}', \rho'_0)$
  e.g. replace MFP with MOP

Challenge

Computation the MOP/MFP solution for cyclic initialized data-flow programs

Accelerated Computation of the MFP Solution (2)

10 $\rho'_0 \leftarrow \rho \circ \kappa$
11 $\rho'' \leftarrow \mathrm{MFP}(\mathcal{S}', \rho'_0)$
12 $\rho \leftarrow \rho \sqcup \kappa^{-1}(\rho'')$

Correctness

$\kappa^{-1}(\rho'') \sqsubseteq \mathrm{MFP}(\mathcal{S}, \rho)$

Alternatives

- line 10: any ρ'_0 such that $\rho'_0 \sqsubseteq \rho \circ \kappa$
  e.g. pick $X' \in \mathcal{X}'$ and define ρ'_0 by $\begin{cases} \rho'_0(X') & = & \rho \circ \kappa(X') \\ \rho'_0(Y') & = & \bot \text{ for all } Y' \neq X' \end{cases}$
- line 11: any ρ'' such that $\rho'' \sqsubseteq \mathrm{MFP}(\mathcal{S}', \rho'_0)$
  e.g. replace MFP with MOP

Challenge

Computation the MOP/MFP solution for cyclic initialized data-flow programs

Outline

Closed Convex Subsets of $\mathbb{R}^n$

Complete Lattice $(A, \sqsubseteq)$

Set of all topologically closed convex subsets of $\mathbb{R}^n$, partially ordered by set inclusion

- greatest lower bound $\sqcap$ is set intersection $\cap$
- least upper bound $\sqcup$ is set union followed by closed convex hull

Closed Convex Polyhedra

$\{\vec{x} \mid M\vec{x} \leq \vec{b}\}$ is called a $\begin{cases} \textit{(real)} \quad \text{polyhedron when } M \in \mathbb{R}^{n \times m} \\ \text{rational polyhedron when } M \in \mathbb{Q}^{n \times m} \end{cases}$

Closed Convex Subsets of $\mathbb{R}^n$

Complete Lattice $(A, \sqsubseteq)$

Set of all topologically closed convex subsets of $\mathbb{R}^n$, partially ordered by set inclusion

- greatest lower bound $\sqcap$ is set intersection $\cap$
- least upper bound $\sqcup$ is set union followed by closed convex hull

Closed Convex Polyhedra

$\{\vec{x} \mid M\vec{x} \leq \vec{b}\}$ is called a $\begin{cases} \textit{(real)} \quad \text{polyhedron when } M \in \mathbb{R}^{n \times m} \\ \text{rational polyhedron when } M \in \mathbb{Q}^{n \times m} \end{cases}$

Guarded Translation Systems

Idea

Guarded commands of the form: $\texttt{if } \vec{x} \in G \texttt{ then } \vec{x} := \vec{x} + \vec{d}$

Definition

An *n*-dim guarded translation is any *single input* transition whose transfer function $f : A \longrightarrow A$ is of the form:

$$f(C) = (G \cap C) + \vec{d} \qquad \text{where} \qquad \begin{cases} G \in A \text{ is the } \textit{guard} \\ \vec{d} \in \mathbb{R}^n \text{ is the } \textit{displacement} \end{cases}$$

Notation : $X' := (G \cap X) + \vec{d}$ is also written $X \xrightarrow{G,\vec{d}} X'$

Definition

An *n*-dim guarded translation system (*GTS*) is any data-flow program over $(A, \sqsubseteq)$ whose transitions are *n*-dim guarded translations

Guarded Translation Systems

Idea

Guarded commands of the form: $\texttt{if } \vec{x} \in G \texttt{ then } \vec{x} := \vec{x} + \vec{d}$

Definition

An *n*-dim guarded translation is any *single input* transition whose transfer function $f : A \to A$ is of the form:

$$f(C) = (G \cap C) + \vec{d} \qquad \text{where} \qquad \begin{cases} G \in A \text{ is the } \textit{guard} \\ \vec{d} \in \mathbb{R}^n \text{ is the } \textit{displacement} \end{cases}$$

Notation : $X' := (G \cap X) + \vec{d}$ is also written $X \xrightarrow{G,\vec{d}} X'$

Definition

An *n*-dim guarded translation system (*GTS*) is any data-flow program over $(A, \sqsubseteq)$ whose transitions are *n*-dim guarded translations

Guarded Translation Systems

Idea

Guarded commands of the form: $\texttt{if } \vec{x} \in G \texttt{ then } \vec{x} := \vec{x} + \vec{d}$

Definition

An *n*-dim guarded translation is any *single input* transition whose transfer function $f : A \to A$ is of the form:

$$f(C) = (G \cap C) + \vec{d} \qquad \text{where} \qquad \begin{cases} G \in A \text{ is the } \textit{guard} \\ \vec{d} \in \mathbb{R}^n \text{ is the } \textit{displacement} \end{cases}$$

Notation : $X' := (G \cap X) + \vec{d}$ is also written $X \xrightarrow{G,\vec{d}} X'$

Definition

An *n*-dim guarded translation system (*GTS*) is any data-flow program over $(A, \sqsubseteq)$ whose transitions are *n*-dim guarded translations

Guarded Translation Systems (Semantics Rephrase)

Definition (Recall)

An *n-dim GTS* is any pair $\mathcal{S} = (\mathcal{X}, T)$ where:

- $\mathcal{X}$ is a finite set of *variables*
- T is a finite set of *n*-dim *guarded translations* $X \xrightarrow{G,\vec{d}} X'$

The complete lattice $(A, \sqsubseteq)$ of closed convex subsets of $\mathbb{R}^n$ is extended to the complete lattice of valuations $(\mathcal{X} \to A, \sqsubseteq)$

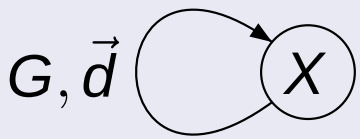
Definition

The data-flow semantics $[\![t]\!]$ of any transition $t = X \xrightarrow{G,\vec{d}} X'$ is the monotonic function in $(\mathcal{X} \to A) \to (\mathcal{X} \to A)$ defined by:

$$\begin{aligned} [\![t]\!](\rho)(X') &= (G \cap \rho(X)) + \vec{d} \\ [\![t]\!](\rho)(Y) &= \rho(Y) \text{ for all } Y \neq X' \end{aligned}$$

Self-Loop GTS Example (2-Dim Closed Convex)

Data-Flow Program

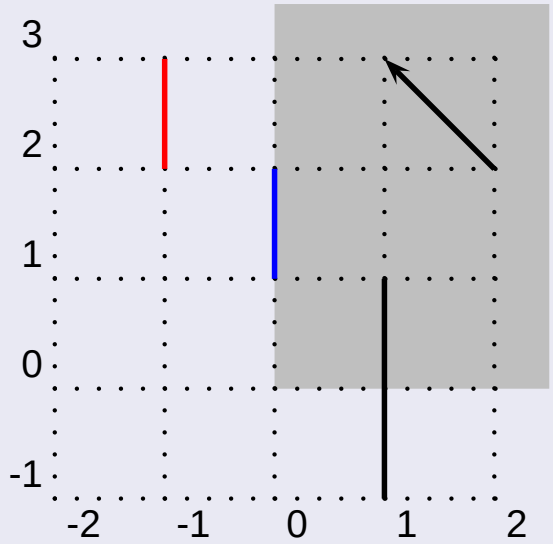


- $\mathcal{X} = \{X\}$
- $T = \{t\}$
- $(t) \quad X := (G \cap X) + \vec{d}$

  with $\begin{cases} G & = & \mathbb{R}^2_+ \\ \vec{d} & = & (-1, 1) \end{cases}$

- $\rho_0 = \{X \mapsto 1 \times [-1, 1]\}$

Semantics



$$\begin{array}{lcl} \rho_1 & = & [\![t]\!](\rho_0) \\ \rho_2 & = & [\![t]\!](\rho_1) \\ \rho_3 & = & [\![t]\!](\rho_2) = \{X \mapsto \emptyset\} \end{array}$$

Instanciating the Acceleration Framework

Challenge

Computation the MOP/MFP solution for cyclic initialized guarded translation systems (IGTS)

- Permits (exact) acceleration of the Kleene fix-point iteration
- Raises new interesting theoretical questions !

Outline

Instanciating the Acceleration Framework

Challenge

Computation the MOP/MFP solution for cyclic initialized guarded translation systems (IGTS)

- Permits (exact) acceleration of the Kleene fix-point iteration
- Raises new interesting theoretical questions !

Outline

1. Acceleration for Self-Loops
2. Acceleration for Cycles

Outline

MOP Solution for Self-Loop IGTS

Theorem

For any n-dim self-loop IGTS $\mathcal{S} = (\{X\}, \{X \xrightarrow{G,\vec{d}} X\}, \rho_0)$*, if* G *and* $\rho_0(X)$ *are polyhedra then* $\mathsf{MOP}(\mathcal{S}, \rho_0)$ *is a polyhedron*

Proof Sketch

$$\mathsf{MOP}(\mathcal{S}, \rho_0)(X) = \rho_0(X) \sqcup \left(\mathrm{cloconv}\left(G \cap \left((G \cap \rho_0(X)) + \mathbb{N}\vec{d}\right)\right) + \vec{d}\right)$$

- Poly-based semilinear subsets of $\mathbb{R}^n$: $\bigcup \left(B + \sum_{\vec{p} \in P} \mathbb{N}\vec{p}\right)$
- Closure of this class under sum, union and intersection
- $\mathrm{cloconv}(S)$ is a polyhedron when S is poly-based semilinear

Remark

The proof is constructive

MOP Solution for Self-Loop IGTS

Theorem

For any n-dim self-loop IGTS $\mathcal{S} = (\{X\}, \{X \xrightarrow{G,\vec{d}} X\}, \rho_0)$*, if* G *and* $\rho_0(X)$ *are polyhedra then* $\mathrm{MOP}(\mathcal{S}, \rho_0)$ *is a polyhedron*

Proof Sketch

$$\mathrm{MOP}(\mathcal{S}, \rho_0)(X) = \rho_0(X) \sqcup \left(\mathrm{cloconv}\left(G \cap \left((G \cap \rho_0(X)) + \mathbb{N}\vec{d}\right)\right) + \vec{d}\right)$$

- Poly-based semilinear subsets of $\mathbb{R}^n$: $\bigcup \left(B + \sum_{\vec{p} \in P} \mathbb{N}\vec{p}\right)$
- Closure of this class under sum, union and intersection
- $\mathrm{cloconv}(S)$ is a polyhedron when S is poly-based semilinear

Remark

The proof is constructive

MFP Solution for Self-Loop IGTS

Theorem

For any n-dim self-loop IGTS $(\{X\}, \{X \xrightarrow{G,\vec{d}} X\}, \rho_0)$*, the MFP solution is the valuation:*

$$X \mapsto \begin{cases} \rho_0(X) & \text{if } G \cap \rho_0(X) = \emptyset \\ \rho_0(X) \sqcup ((G \cap (\rho_0(X) + \mathbb{R}_+\vec{d})) + \vec{d}) & \text{otherwise} \end{cases}$$

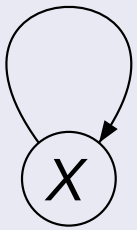
Proof Ideas

- $\subseteq$ The given expression is a post-fix-point of $\left[\!\left[X \xrightarrow{G,\vec{d}} X\right]\!\right]$.
- $\supseteq$ Proof by contradiction, using topological and convexity properties of both the guard and MFP solution.

Comparison with Standard Widening on Polyhedra [Cousot & Halbwachs, POPL'78]

IGTS

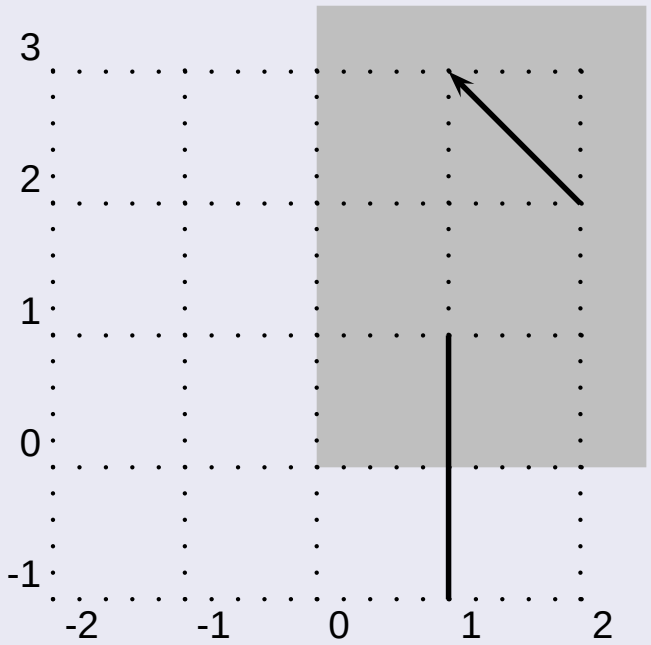
$$\mathbb{R}^2_+, (-1, 1)$$



$$\rho_0 = \{X \mapsto 1 \times [-1, 1]\}$$

- Application of widening
- Coarser than the MFP Solution!

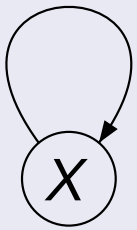
Iteration with Widening



Comparison with Standard Widening on Polyhedra [Cousot & Halbwachs, POPL'78]

IGTS

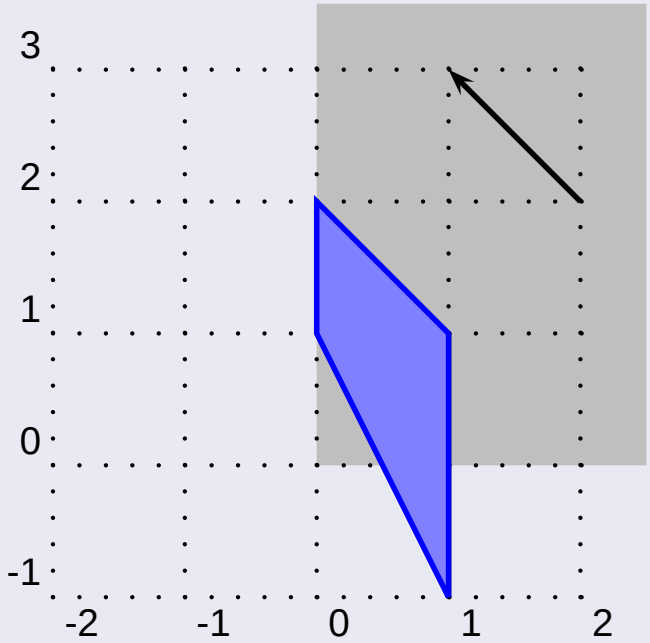
$$\mathbb{R}^2_+, (-1, 1)$$



$$\rho_0 = \{X \mapsto 1 \times [-1, 1]\}$$

- Application of widening
- Coarser than the MFP Solution!

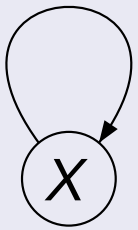
Iteration with Widening



Comparison with Standard Widening on Polyhedra [Cousot & Halbwachs, POPL'78]

IGTS

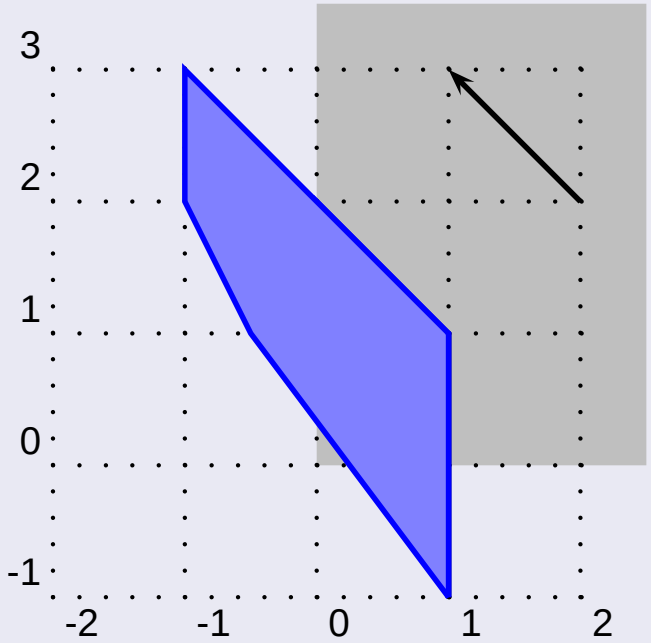
$$\mathbb{R}^2_+, (-1, 1)$$



$$\rho_0 = \{X \mapsto 1 \times [-1, 1]\}$$

- Application of widening
- Coarser than the MFP Solution!

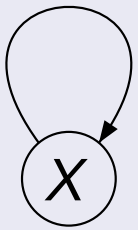
Iteration with Widening



Comparison with Standard Widening on Polyhedra [Cousot & Halbwachs, POPL'78]

IGTS

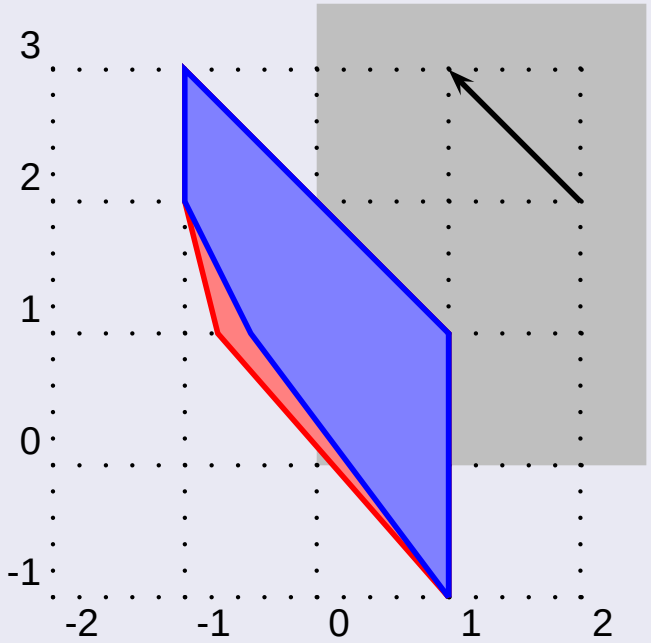
$$\mathbb{R}^2_+, (-1, 1)$$



$$\rho_0 = \{X \mapsto 1 \times [-1, 1]\}$$

- Application of widening
- Coarser than the MFP Solution!

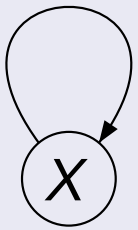
Iteration with Widening



Comparison with Standard Widening on Polyhedra [Cousot & Halbwachs, POPL'78]

IGTS

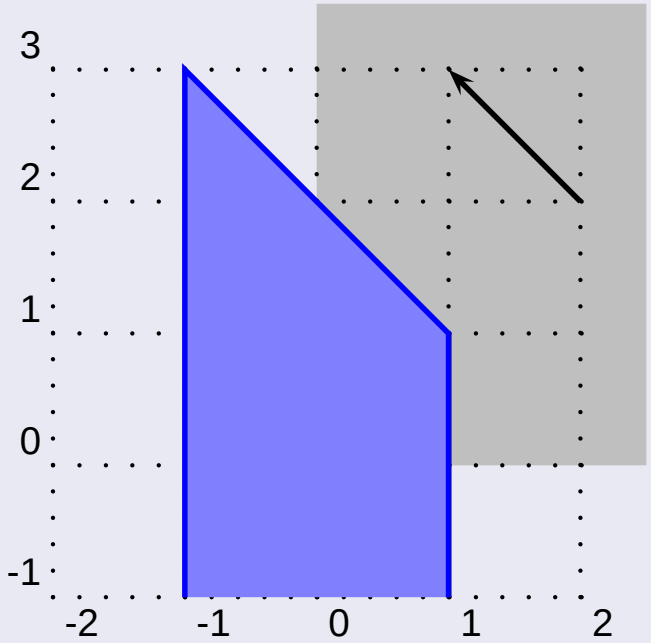
$$\mathbb{R}^2_+, (-1, 1)$$



$$\rho_0 = \{X \mapsto 1 \times [-1, 1]\}$$

- Application of widening
- Coarser than the MFP Solution!

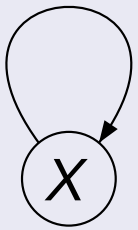
Iteration with Widening



Comparison with Standard Widening on Polyhedra [Cousot & Halbwachs, POPL'78]

IGTS

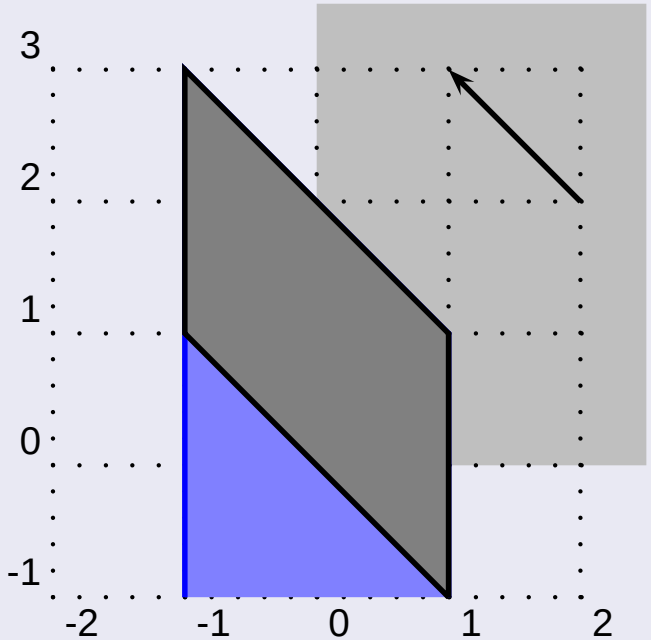
$$\mathbb{R}^2_+, (-1, 1)$$



$$\rho_0 = \{X \mapsto 1 \times [-1, 1]\}$$

- Application of widening
- Coarser than the MFP Solution!

Iteration with Widening



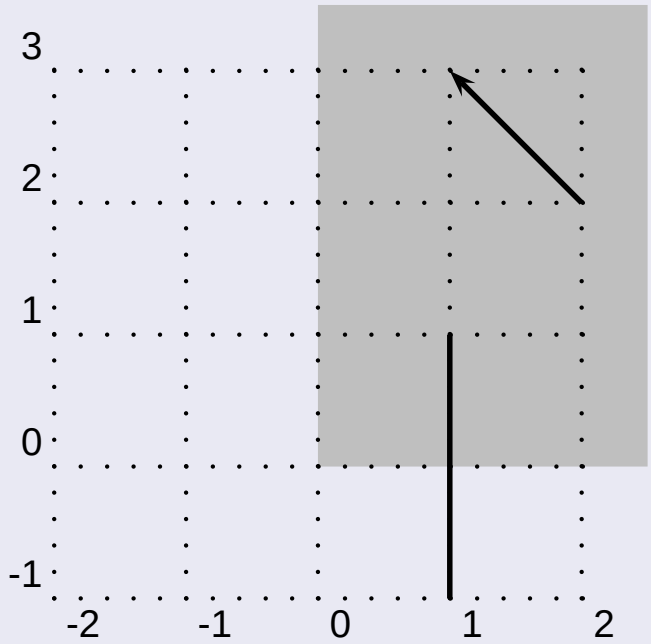
Comparison with Polyhedral Abstract Acceleration [Gonnord & Halbwachs, SAS'06]

- Consider an IGTS $\mathcal{S} = (\{X\}, \{X \xrightarrow{G,\vec{d}} X\}, \rho_0)$

Abstract Acceleration

AbAc($\mathcal{S}$) $= \rho_0(X) \sqcup \text{MFP}(\mathcal{S}')$ where $\mathcal{S}'$ is equal to $\mathcal{S}$ except on its initial valuation: $\rho'_0(X) = G \cap \rho_0(X)$.

Iteration with Abs. Acc.



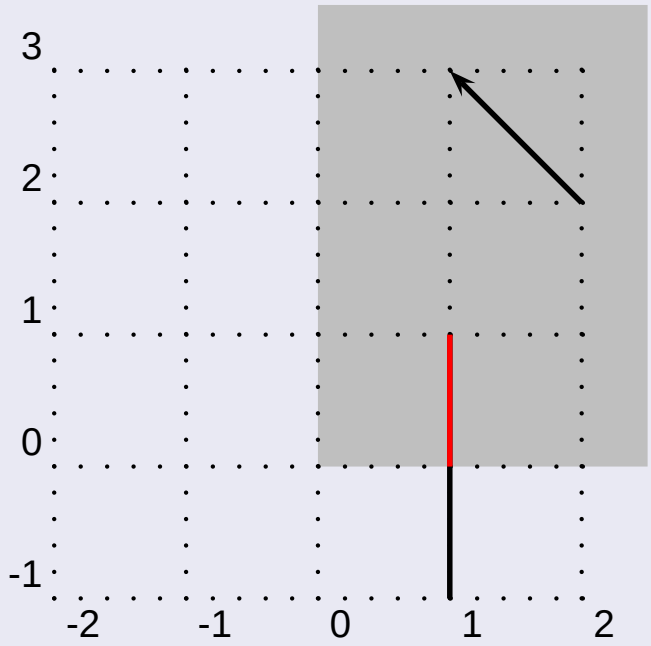
Comparison with Polyhedral Abstract Acceleration [Gonnord & Halbwachs, SAS'06]

- Consider an IGTS $\mathcal{S} = (\{X\}, \{X \xrightarrow{G,\vec{d}} X\}, \rho_0)$

Abstract Acceleration

AbAc($\mathcal{S}$) $= \rho_0(X) \sqcup \text{MFP}(\mathcal{S}')$ where $\mathcal{S}'$ is equal to $\mathcal{S}$ except on its initial valuation: $\rho'_0(X) = G \cap \rho_0(X)$.

Iteration with Abs. Acc.



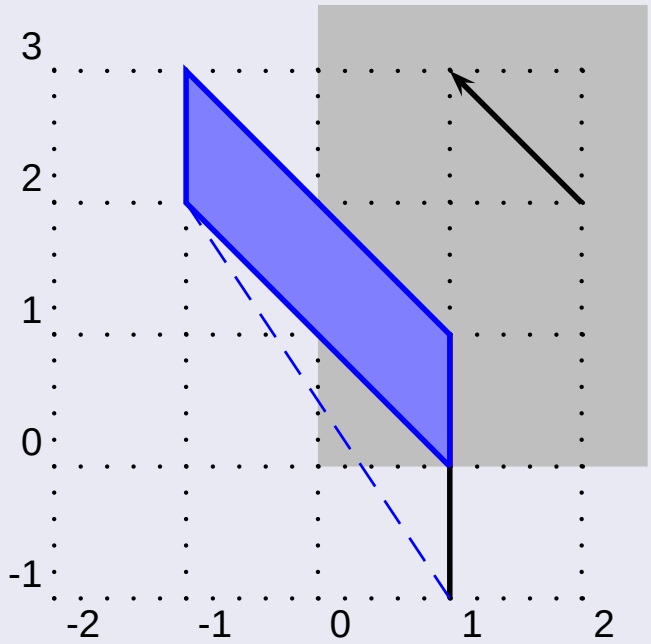
Comparison with Polyhedral Abstract Acceleration [Gonnord & Halbwachs, SAS'06]

- Consider an IGTS $\mathcal{S} = (\{X\}, \{X \xrightarrow{G,\vec{d}} X\}, \rho_0)$

Abstract Acceleration

AbAc($\mathcal{S}$) $= \rho_0(X) \sqcup \text{MFP}(\mathcal{S}')$ where $\mathcal{S}'$ is equal to $\mathcal{S}$ except on its initial valuation: $\rho'_0(X) = G \cap \rho_0(X)$.

Iteration with Abs. Acc.



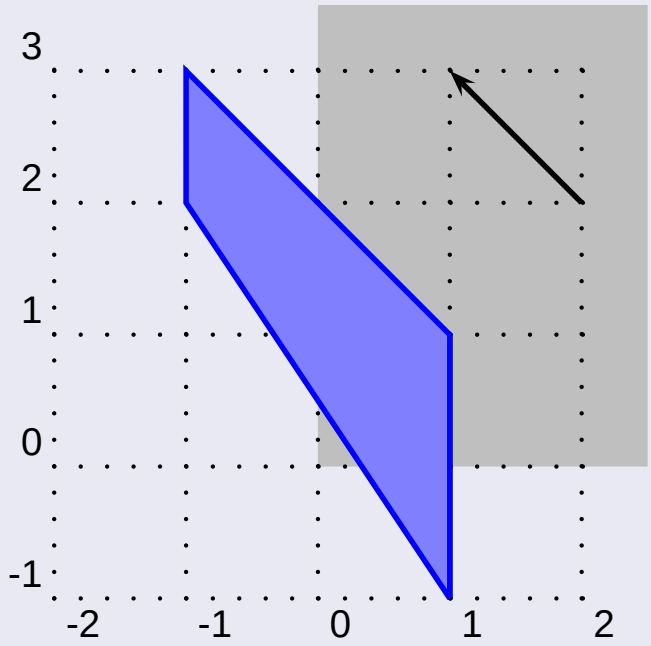
Comparison with Polyhedral Abstract Acceleration [Gonnord & Halbwachs, SAS'06]

- Consider an IGTS $\mathcal{S} = (\{X\}, \{X \xrightarrow{G,\vec{d}} X\}, \rho_0)$

Abstract Acceleration

AbAc($\mathcal{S}$) $= \rho_0(X) \sqcup \text{MFP}(\mathcal{S}')$ where $\mathcal{S}'$ is equal to $\mathcal{S}$ except on its initial valuation: $\rho'_0(X) = G \cap \rho_0(X)$.

Iteration with Abs. Acc.



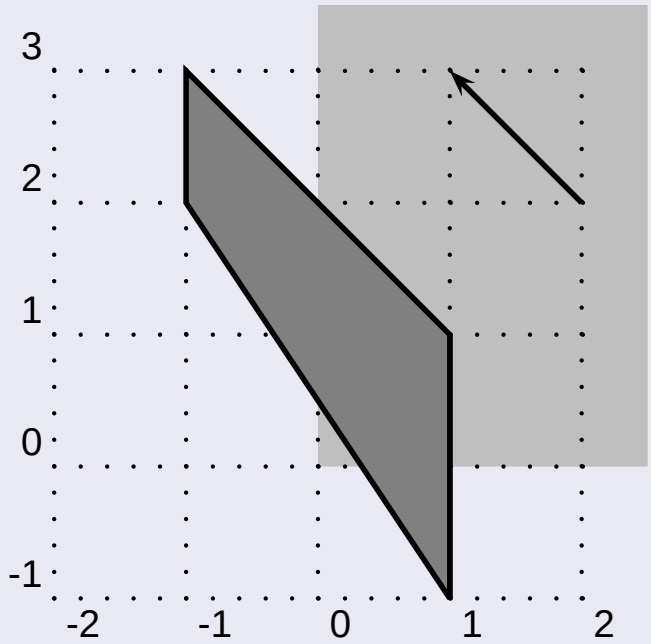
Comparison with Polyhedral Abstract Acceleration [Gonnord & Halbwachs, SAS'06]

- Consider an IGTS $\mathcal{S} = (\{X\}, \{X \xrightarrow{G,\vec{d}} X\}, \rho_0)$

Abstract Acceleration

AbAc($\mathcal{S}$) $= \rho_0(X) \sqcup \text{MFP}(\mathcal{S}')$ where $\mathcal{S}'$ is equal to $\mathcal{S}$ except on its initial valuation: $\rho'_0(X) = G \cap \rho_0(X)$.

Iteration with Abs. Acc.



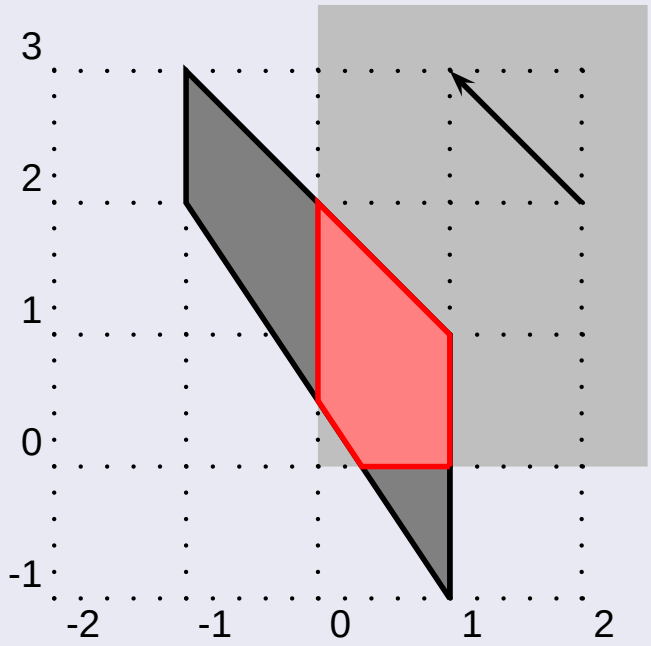
Comparison with Polyhedral Abstract Acceleration [Gonnord & Halbwachs, SAS'06]

- Consider an IGTS $\mathcal{S} = (\{X\}, \{X \xrightarrow{G,\vec{d}} X\}, \rho_0)$

Abstract Acceleration

AbAc($\mathcal{S}$) $= \rho_0(X) \sqcup \text{MFP}(\mathcal{S}')$ where $\mathcal{S}'$ is equal to $\mathcal{S}$ except on its initial valuation: $\rho'_0(X) = G \cap \rho_0(X)$.

Iteration with Abs. Acc.



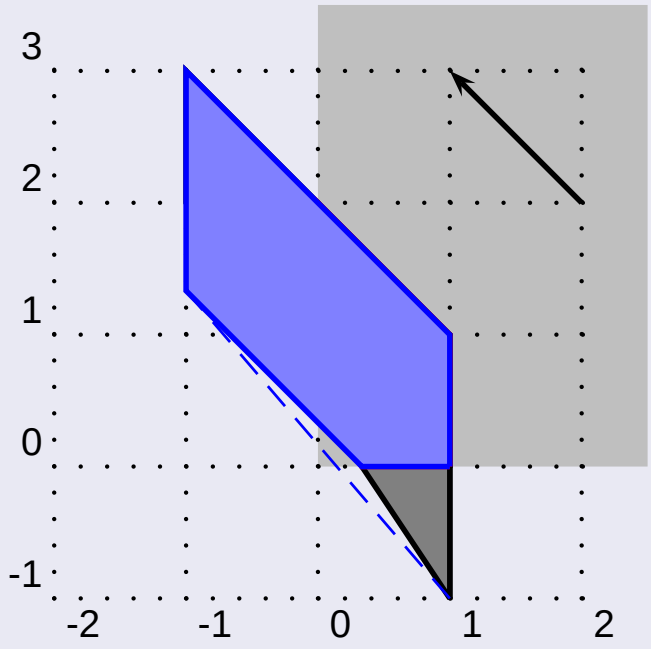
Comparison with Polyhedral Abstract Acceleration [Gonnord & Halbwachs, SAS'06]

- Consider an IGTS $\mathcal{S} = (\{X\}, \{X \xrightarrow{G,\vec{d}} X\}, \rho_0)$

Abstract Acceleration

AbAc($\mathcal{S}$) $= \rho_0(X) \sqcup \text{MFP}(\mathcal{S}')$ where $\mathcal{S}'$ is equal to $\mathcal{S}$ except on its initial valuation: $\rho'_0(X) = G \cap \rho_0(X)$.

Iteration with Abs. Acc.



Comparison with Polyhedral Abstract Acceleration [Gonnord & Halbwachs, SAS'06]

- Consider an IGTS $\mathcal{S} = (\{X\}, \{X \xrightarrow{G,\vec{d}} X\}, \rho_0)$

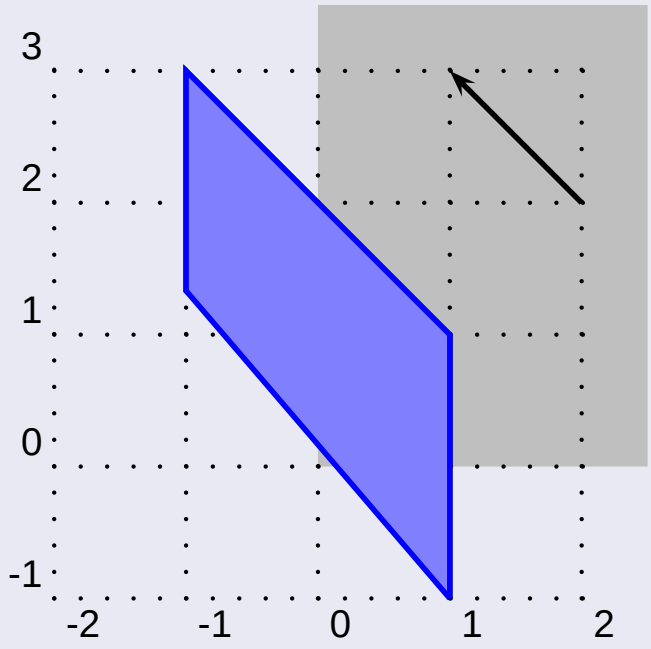
Abstract Acceleration

AbAc($\mathcal{S}$) $= \rho_0(X) \sqcup \text{MFP}(\mathcal{S}')$ where $\mathcal{S}'$ is equal to $\mathcal{S}$ except on its initial valuation: $\rho'_0(X) = G \cap \rho_0(X)$.

Remark

Iteration does not terminate!

Iteration with Abs. Acc.



Outline

MOP Solution for Cyclic IGTS

Consider a cyclic IGTS $\mathcal{S} = (\{X_1, \ldots, X_k\}, \{t_1, \ldots, t_k\}, \rho_0)$ with $t_i = X_i \xrightarrow{G_i, \vec{d}_i} X_{i+1}$ and $X_{k+1} = X_1$, i.e. $X_1 \xrightarrow{G_1, \vec{d}_1} X_2 \cdots X_k \xrightarrow{G_k, \vec{d}_k} X_1$

Let $\mathcal{S}' = (\{X_1\}, \{X_1 \xrightarrow{G, \vec{d}} X_1\})$, where :

$$\begin{cases} G & = & G_1 \cap (G_2 - \vec{d}_1) \cap \cdots \cap (G_k - (\vec{d}_1 + \cdots + \vec{d}_{k-1})) \\ \vec{d} & = & \vec{d}_1 + \cdots + \vec{d}_k \end{cases}$$

The transition $X_1 \xrightarrow{G, \vec{d}} X_1$ "simulates" the cycle $t_1 \cdots t_k$ w.r.t. to X_1

Reduction to the Self-Loop Case

$$\mathrm{MOP}(\mathcal{S}, \rho_0)(X_1) = \bigsqcup_{i=1}^{k-1} \mathrm{MOP}(\mathcal{S}', \{X_1 \mapsto (\llbracket t_k \rrbracket \circ \cdots \circ \llbracket t_{i+1} \rrbracket (\rho_0))(X_1)\})$$

MOP Solution for Cyclic IGTS

Consider a cyclic IGTS $\mathcal{S} = (\{X_1, \ldots, X_k\}, \{t_1, \ldots, t_k\}, \rho_0)$ with $t_i = X_i \xrightarrow{G_i, \vec{d}_i} X_{i+1}$ and $X_{k+1} = X_1$, i.e. $X_1 \xrightarrow{G_1, \vec{d}_1} X_2 \cdots X_k \xrightarrow{G_k, \vec{d}_k} X_1$

Let $\mathcal{S}' = (\{X_1\}, \{X_1 \xrightarrow{G, \vec{d}} X_1\})$, where :

$$\begin{cases} G & = & G_1 \cap (G_2 - \vec{d}_1) \cap \cdots \cap (G_k - (\vec{d}_1 + \cdots + \vec{d}_{k-1})) \\ \vec{d} & = & \vec{d}_1 + \cdots + \vec{d}_k \end{cases}$$

The transition $X_1 \xrightarrow{G, \vec{d}} X_1$ "simulates" the cycle $t_1 \cdots t_k$ w.r.t. to X_1

Reduction to the Self-Loop Case

$$\mathrm{MOP}(\mathcal{S}, \rho_0)(X_1) = \bigsqcup_{i=1}^{k-1} \mathrm{MOP}(\mathcal{S}', \{X_1 \mapsto (\llbracket t_k \rrbracket \circ \cdots \circ \llbracket t_{i+1} \rrbracket (\rho_0))(X_1)\})$$

MFP Solution for Singly-Initialized Cyclic IGTS

Consider a cyclic IGTS $\mathcal{S} = (\{X_1, \ldots, X_k\}, \{t_1, \ldots, t_k\}, \rho_0)$ with $t_i = X_i \xrightarrow{G_i, \vec{d}_i} X_{i+1}$ and $X_{k+1} = X_1$, i.e. $X_1 \xrightarrow{G_1, \vec{d}_1} X_2 \cdots X_k \xrightarrow{G_k, \vec{d}_k} X_1$

Let $\mathcal{S}' = (\{X_1\}, \{X_1 \xrightarrow{G, \vec{d}} X_1\})$, where :

$$\begin{cases} G &= G_1 \cap (G_2 - \vec{d}_1) \cap \cdots \cap (G_k - (\vec{d}_1 + \cdots + \vec{d}_{k-1})) \\ \vec{d} &= \vec{d}_1 + \cdots + \vec{d}_k \end{cases}$$

The transition $X_1 \xrightarrow{G, \vec{d}} X_1$ "simulates" the cycle $t_1 \cdots t_k$ w.r.t. to X_1

Reduction to the Self-Loop Case

If $\rho(Y) = \bot$ for all $Y \neq X_1$ then

$$\mathrm{MFP}(\mathcal{S}, \rho_0) \;=\; [\![t_{k-1}]\!] \circ \cdots \circ [\![t_1]\!](\mathrm{MFP}(\mathcal{S}', \{X_1 \mapsto \rho_0(X_1)\}))$$

MFP Solution for Singly-Initialized Cyclic IGTS

Consider a cyclic IGTS $\mathcal{S} = (\{X_1, \ldots, X_k\}, \{t_1, \ldots, t_k\}, \rho_0)$ with $t_i = X_i \xrightarrow{G_i, \vec{d}_i} X_{i+1}$ and $X_{k+1} = X_1$, i.e. $X_1 \xrightarrow{G_1, \vec{d}_1} X_2 \cdots X_k \xrightarrow{G_k, \vec{d}_k} X_1$

Let $\mathcal{S}' = (\{X_1\}, \{X_1 \xrightarrow{G, \vec{d}} X_1\})$, where :

$$\begin{cases} G &= G_1 \cap (G_2 - \vec{d}_1) \cap \cdots \cap (G_k - (\vec{d}_1 + \cdots + \vec{d}_{k-1})) \\ \vec{d} &= \vec{d}_1 + \cdots + \vec{d}_k \end{cases}$$

The transition $X_1 \xrightarrow{G, \vec{d}} X_1$ "simulates" the cycle $t_1 \cdots t_k$ w.r.t. to X_1

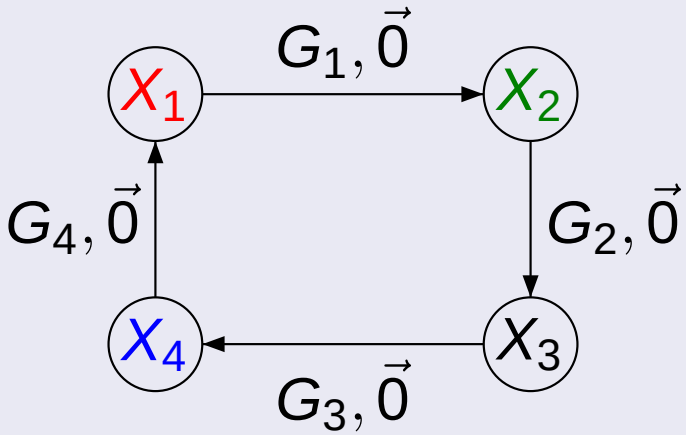
Reduction to the Self-Loop Case

If $\rho(Y) = \bot$ for all $Y \neq X_1$ then

$$\text{MFP}(\mathcal{S}, \rho_0) \;=\; [\![t_{k-1}]\!] \circ \cdots \circ [\![t_1]\!](\text{MFP}(\mathcal{S}', \{X_1 \mapsto \rho_0(X_1)\}))$$

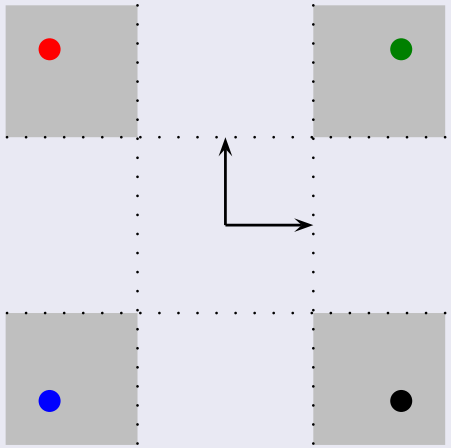
2-dim Cyclic Example

GTS



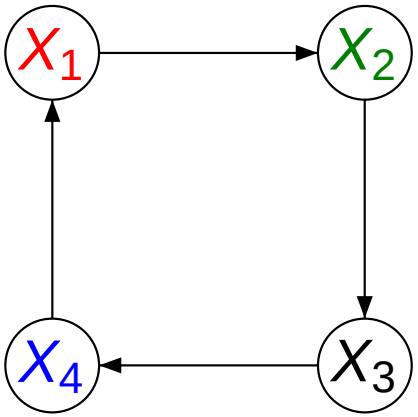
$$
\begin{aligned}
G_1 &=]-\infty,-1] \times [1,+\infty[\\
G_2 &= [1,+\infty[\times [1,+\infty[\\
G_3 &= [1,+\infty[\times]-\infty,-1] \\
G_4 &=]-\infty,-1] \times]-\infty,-1]
\end{aligned}
$$

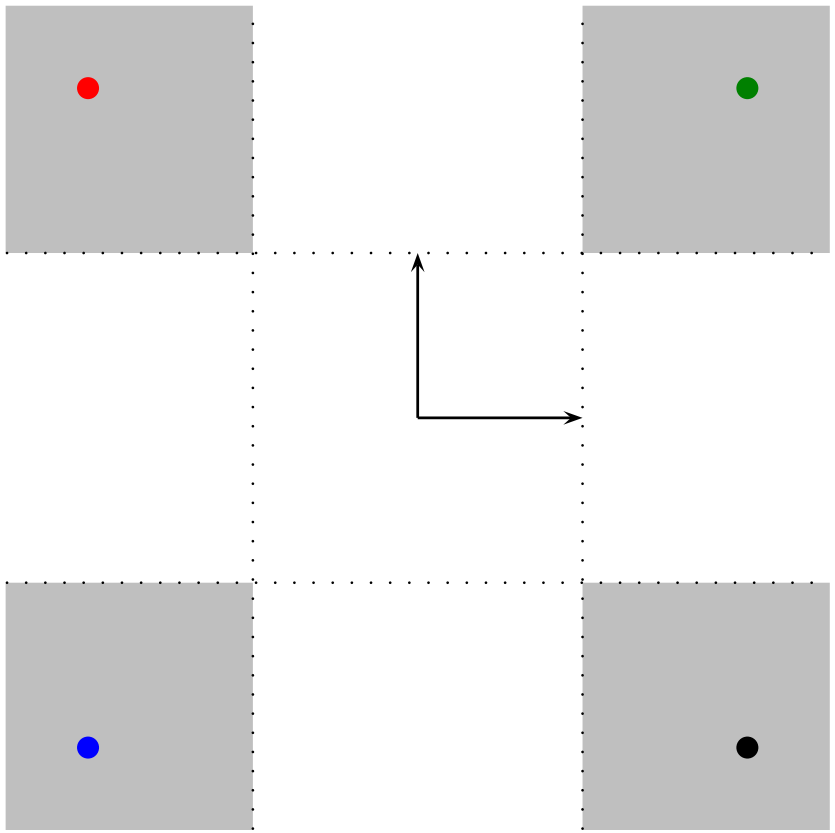
Initial Valuation



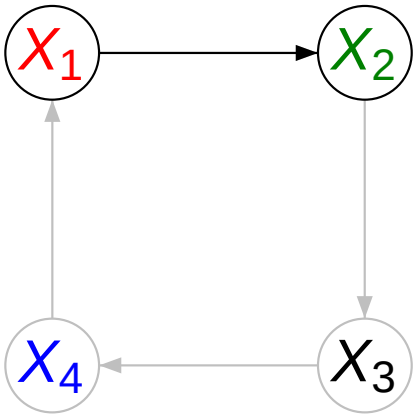
$$
\begin{aligned}
X_1 &\mapsto \{(-2,2)\} \\
X_2 &\mapsto \{(2,2)\} \\
X_3 &\mapsto \{(2,-2)\} \\
X_4 &\mapsto \{(-2,-2)\}
\end{aligned}
$$

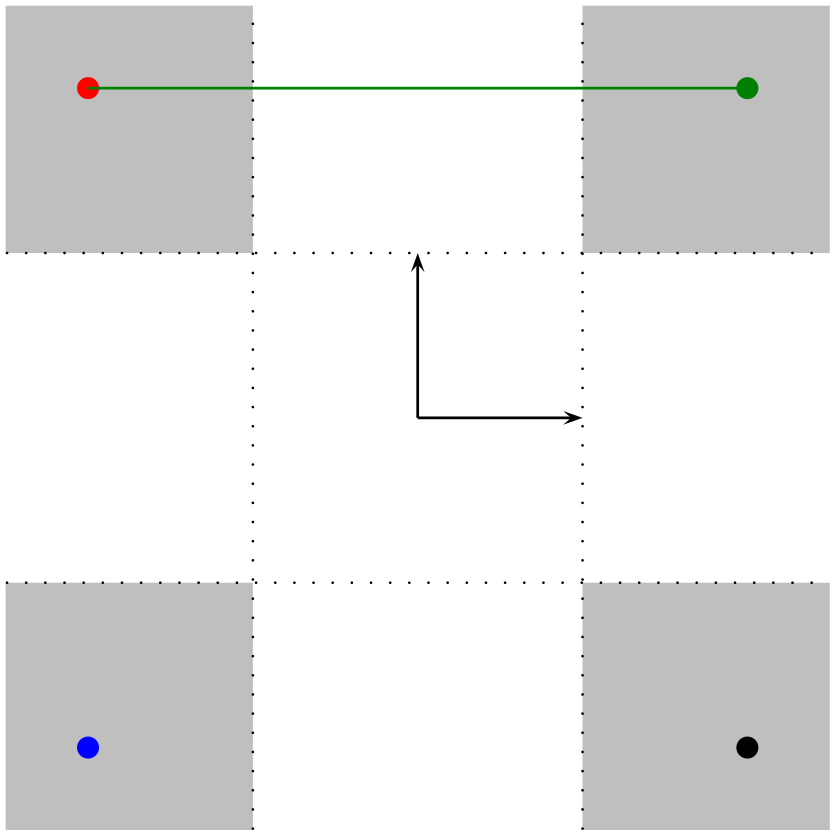
Kleene iteration on 2-dim Cyclic Example



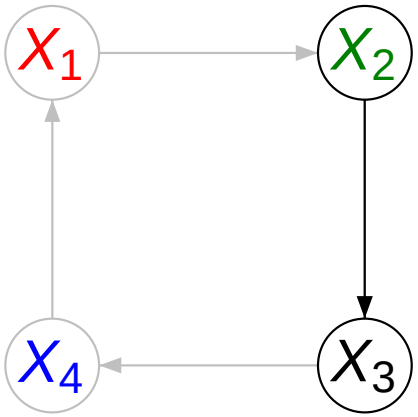


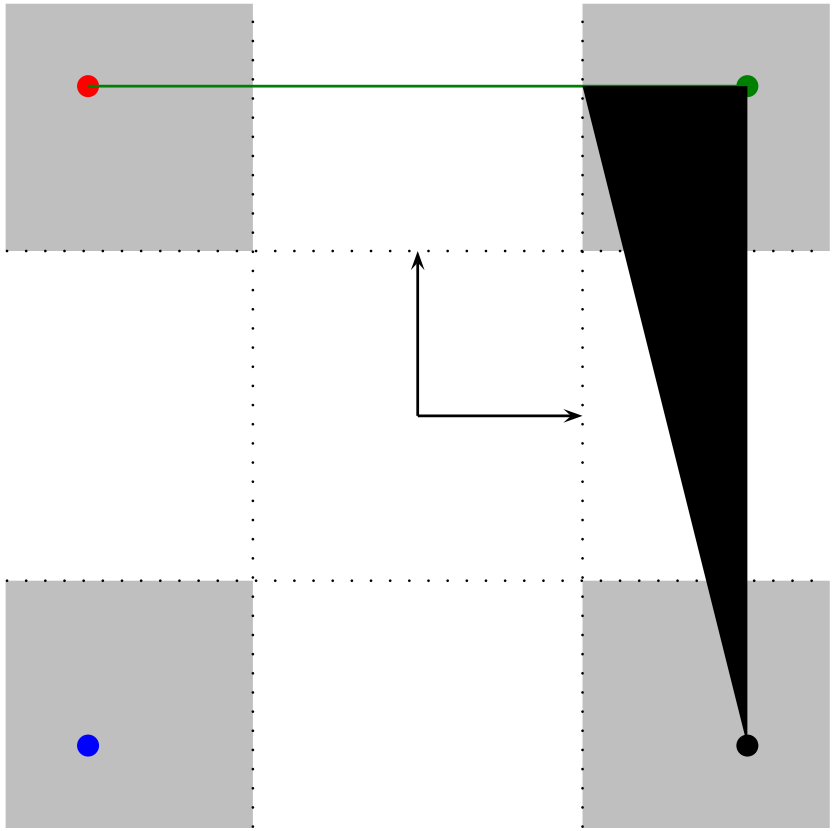
Kleene iteration on 2-dim Cyclic Example



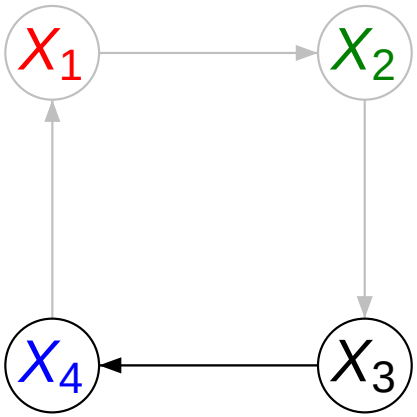


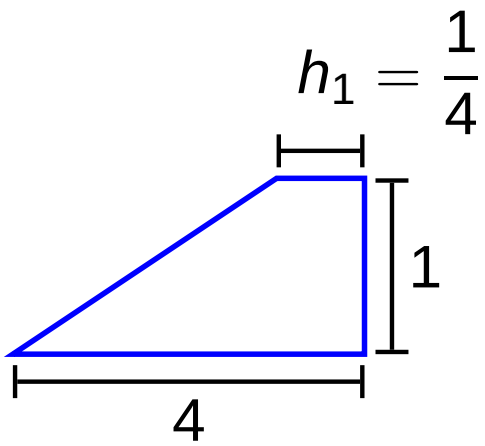
Kleene iteration on 2-dim Cyclic Example

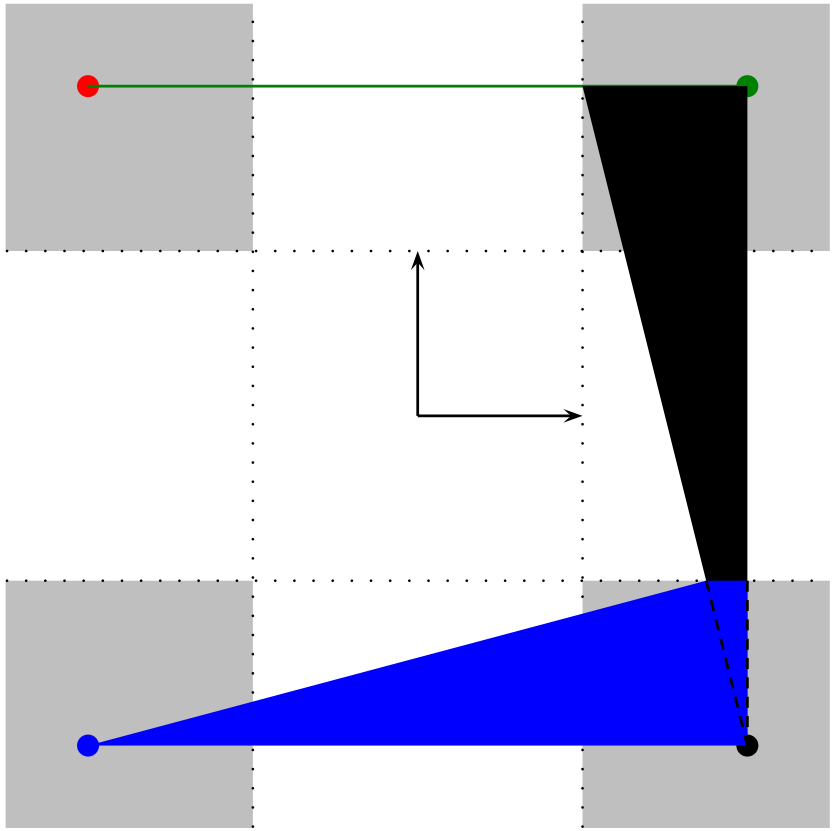




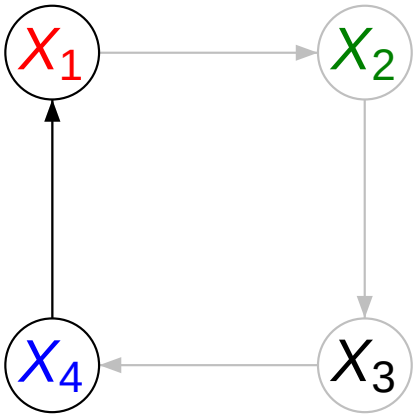
Kleene iteration on 2-dim Cyclic Example

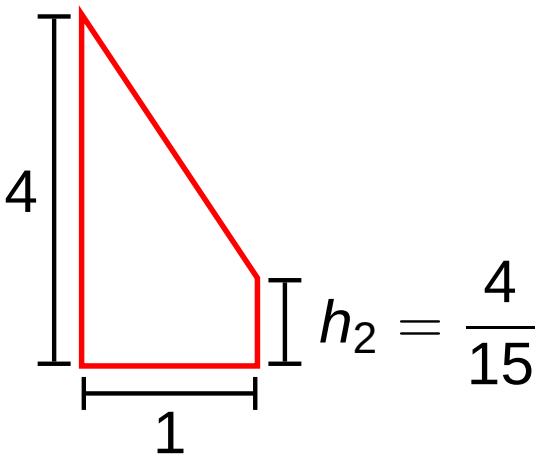


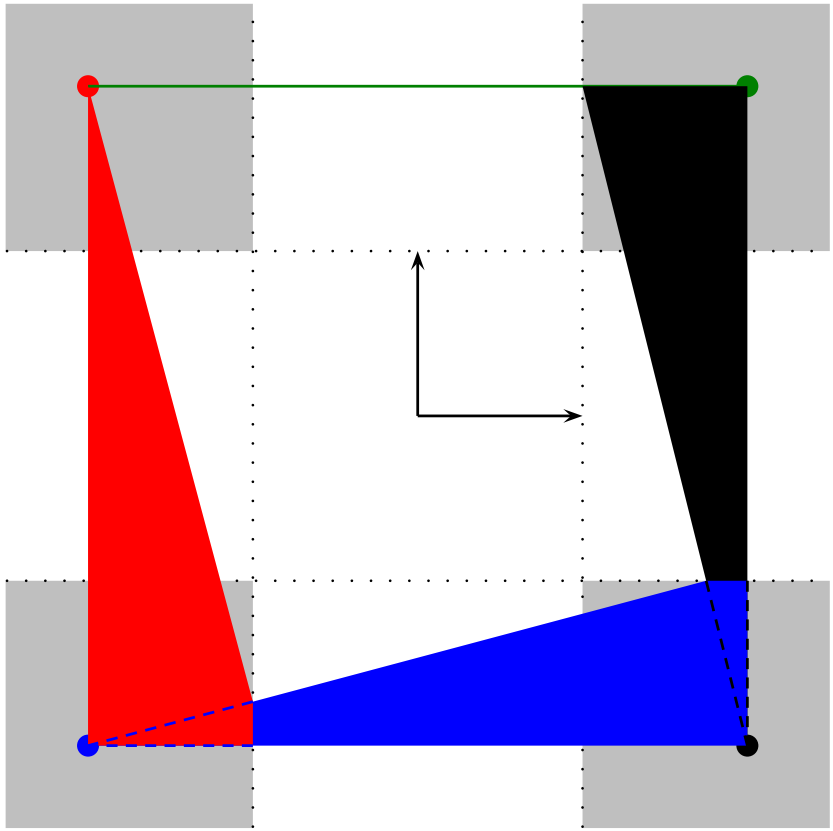




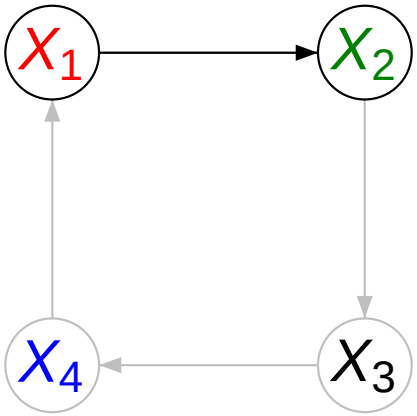
Kleene iteration on 2-dim Cyclic Example

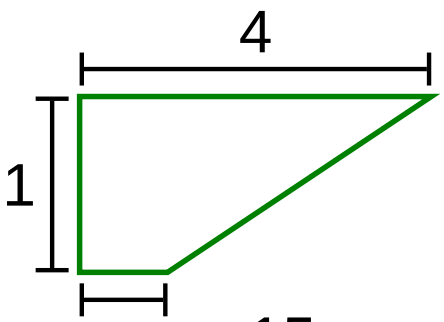




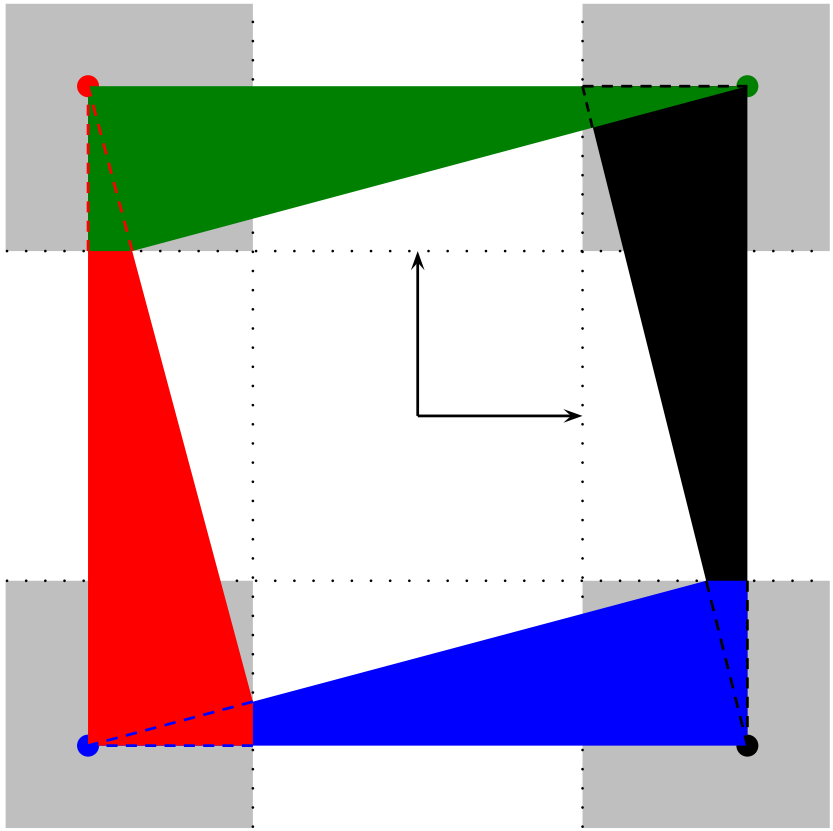


Kleene iteration on 2-dim Cyclic Example

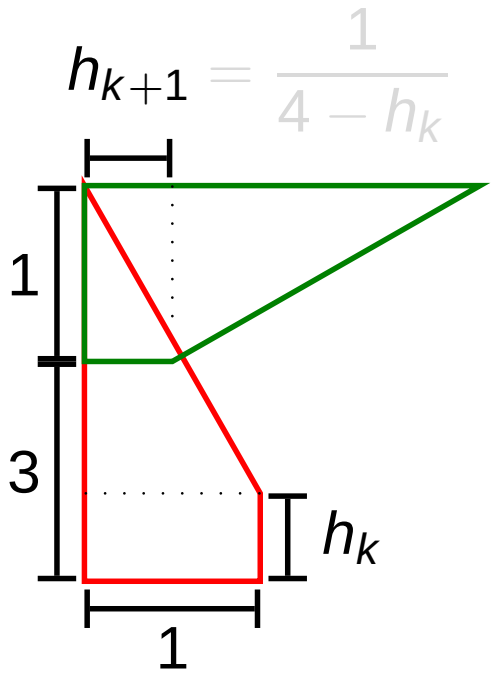


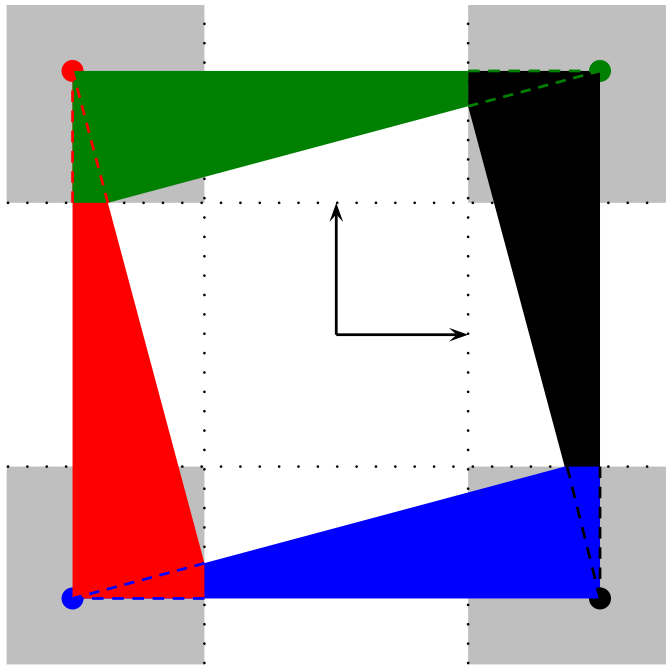


$$h_3 = \frac{15}{56}$$



MFP Solution for 2-dim Example



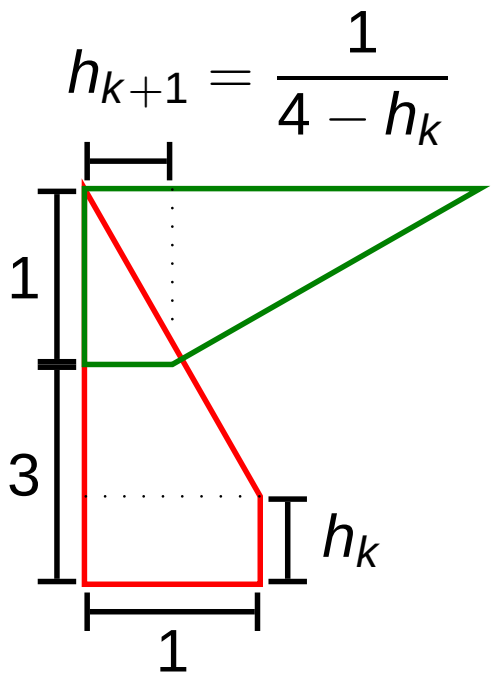


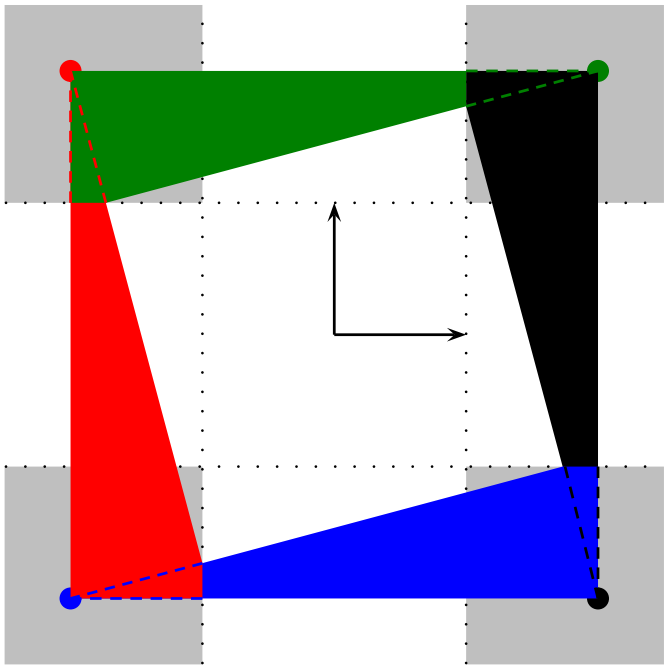
- $(h_k)_{k \in \mathbb{N}}$ is nondecreasing, and $\lim_{k \to \infty} h_k = 2 - \sqrt{3}$

Remark

The MFP solution of this 2-dim cyclic IGTS is not rational polyhedral

MFP Solution for 2-dim Example



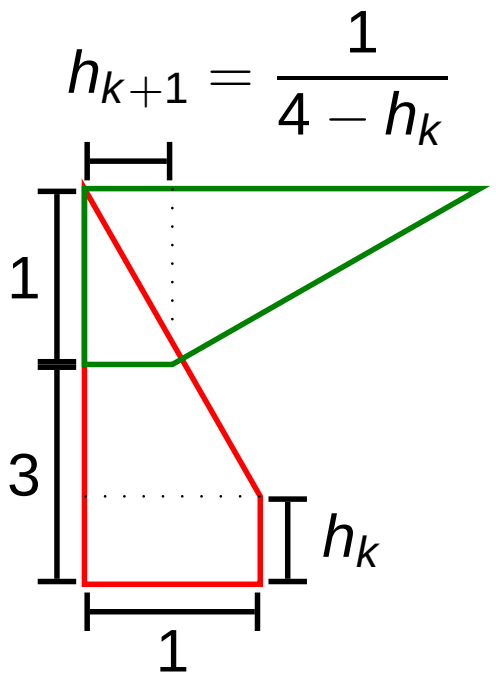


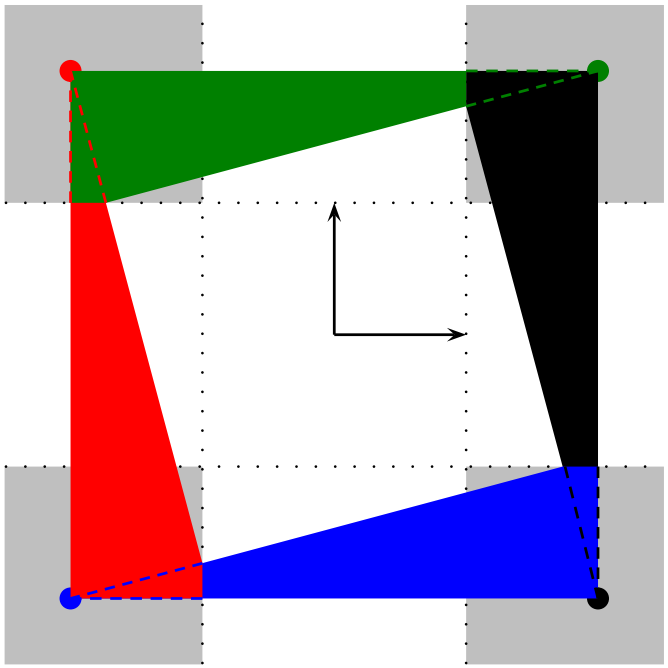
- $(h_k)_{k \in \mathbb{N}}$ is nondecreasing, and $\lim_{k \to \infty} h_k = 2 - \sqrt{3}$

Remark

The MFP solution of this 2-dim cyclic IGTS is not rational polyhedral

MFP Solution for 2-dim Example



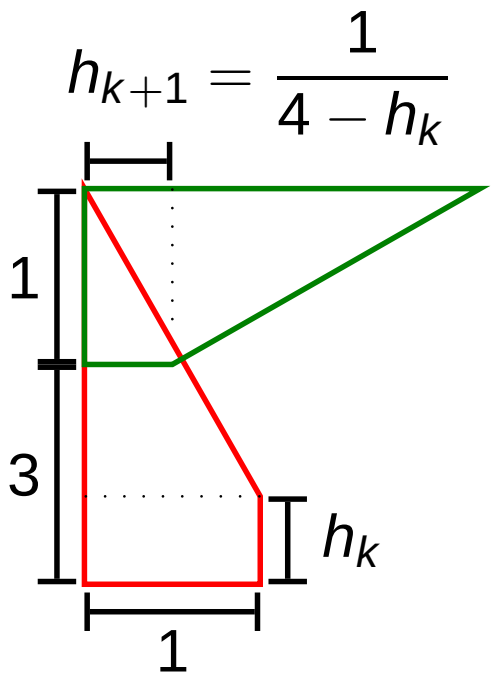


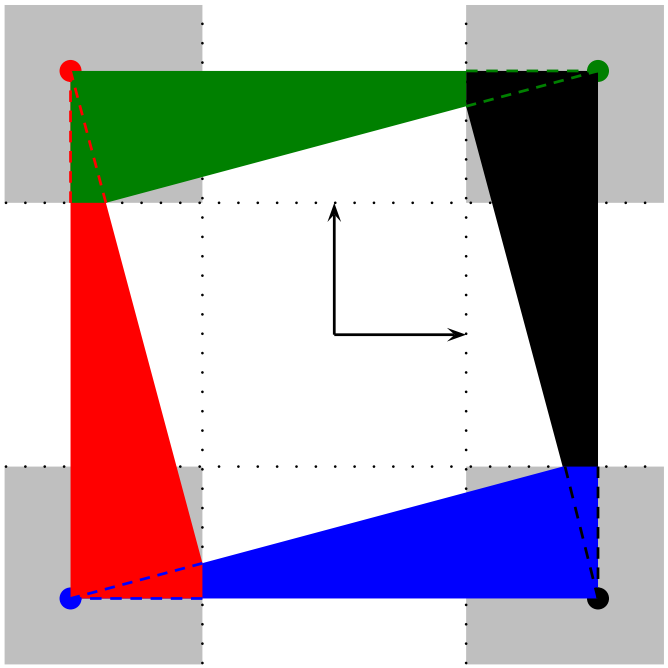
- $(h_k)_{k \in \mathbb{N}}$ is nondecreasing, and $\lim_{k \to \infty} h_k = 2 - \sqrt{3}$

Remark

The MFP solution of this 2-dim cyclic IGTS is not rational polyhedral

MFP Solution for 2-dim Example





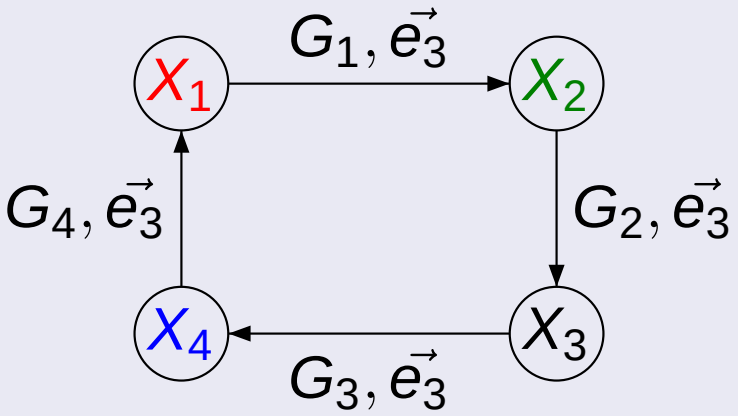
- $(h_k)_{k \in \mathbb{N}}$ is nondecreasing, and $\lim_{k \to \infty} h_k = 2 - \sqrt{3}$

Remark

The MFP solution of this 2-dim cyclic IGTS is not rational polyhedral

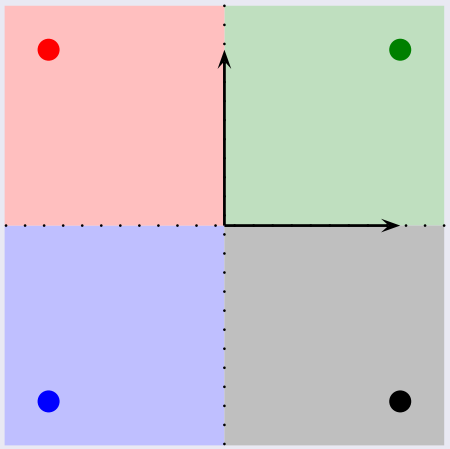
3-dim Cyclic Example

GTS



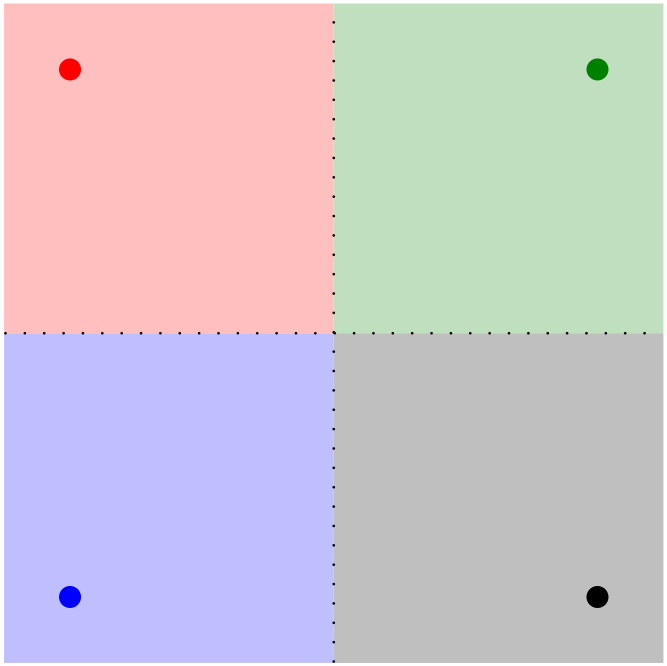
$$
\begin{aligned}
G_1 &= \mathbb{R}_- \times \mathbb{R}^+ \times \mathbb{R} \\
G_2 &= \mathbb{R}_+ \times \mathbb{R}_+ \times \mathbb{R} \\
G_3 &= \mathbb{R}^+ \times \mathbb{R}_- \times \mathbb{R} \\
G_4 &= \mathbb{R}^- \times \mathbb{R}^- \times \mathbb{R}
\end{aligned}
$$

Initial Valuation



$$
\begin{aligned}
X_1 &\mapsto \{(-1,1)\} \times \mathbb{R}^+ \\
X_2 &\mapsto \{(1,1)\} \times \mathbb{R}^+ \\
X_3 &\mapsto \{(1,-1)\} \times \mathbb{R}^+ \\
X_4 &\mapsto \{(-1,-1)\} \times \mathbb{R}^+
\end{aligned}
$$

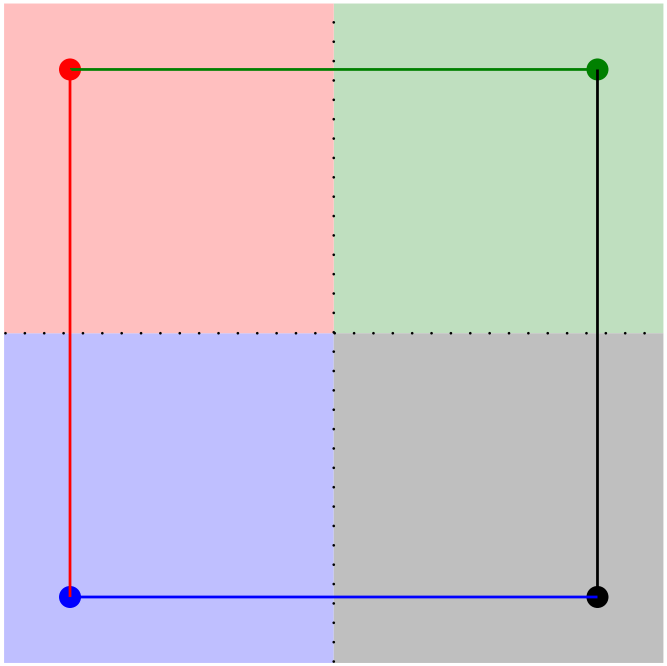
Kleene iteration on 3-dim Cyclic Example



Remark

The MFP solution of this 3-dim cyclic IGTS is not polyhedral

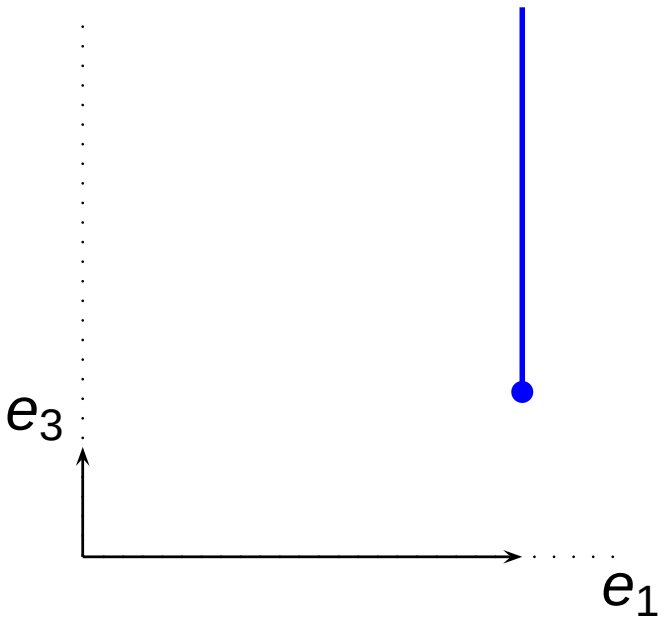
Kleene iteration on 3-dim Cyclic Example

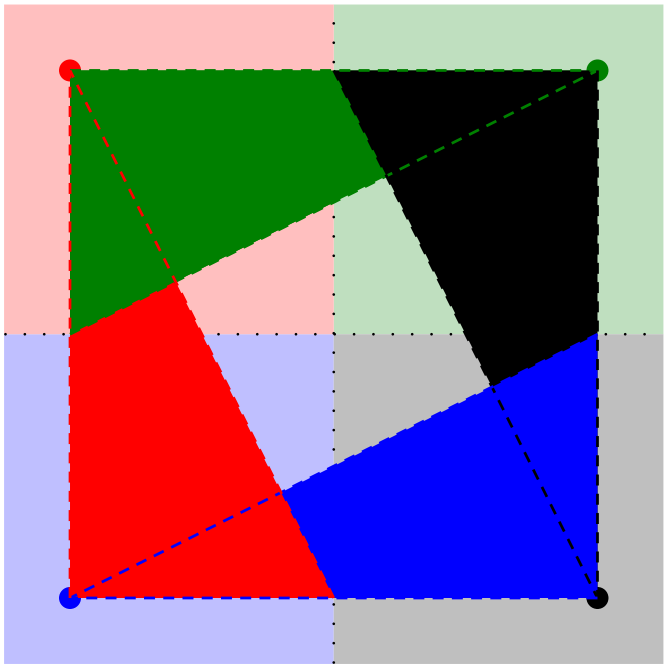


Remark

The MFP solution of this 3-dim cyclic IGTS is not polyhedral

Kleene iteration on 3-dim Cyclic Example

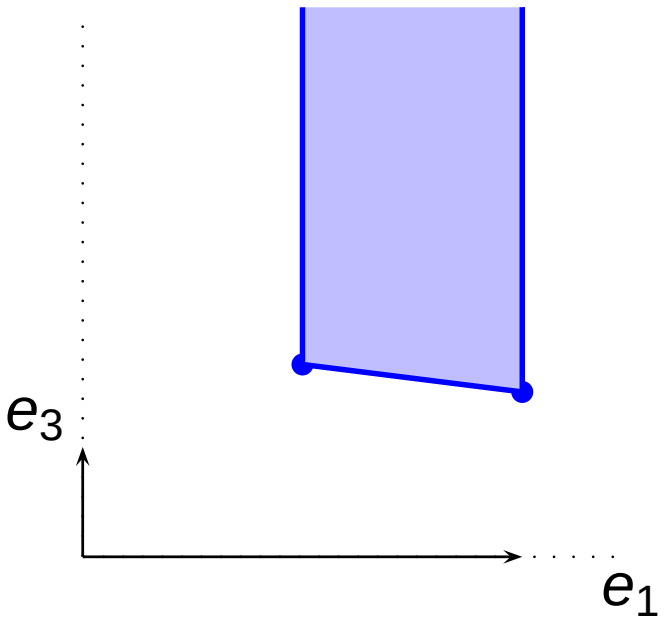


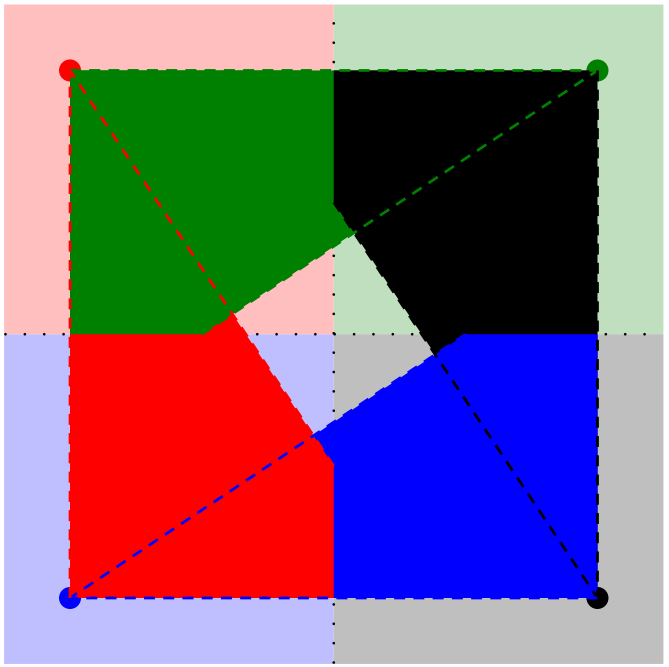


Remark

The MFP solution of this 3-dim cyclic IGTS is not polyhedral

Kleene iteration on 3-dim Cyclic Example

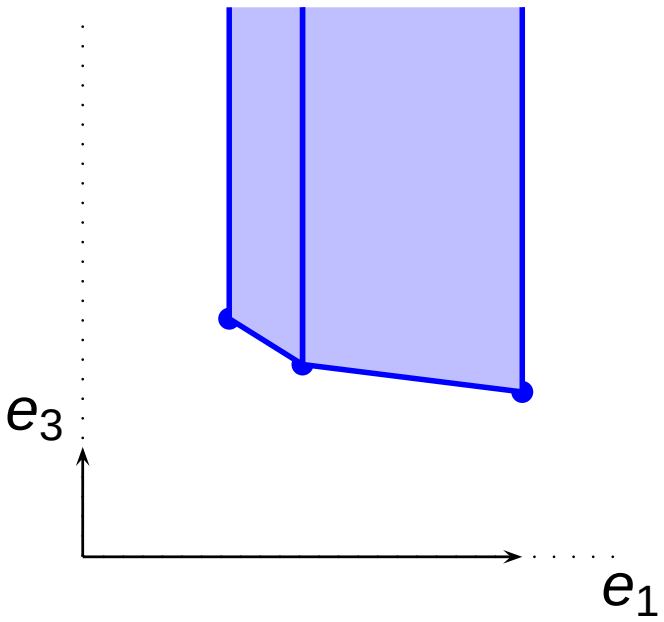


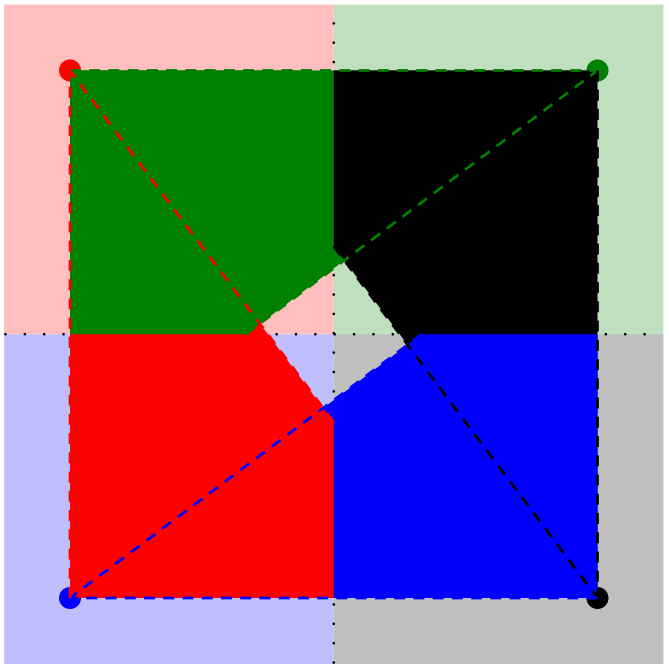


Remark

The MFP solution of this 3-dim cyclic IGTS is not polyhedral

Kleene iteration on 3-dim Cyclic Example

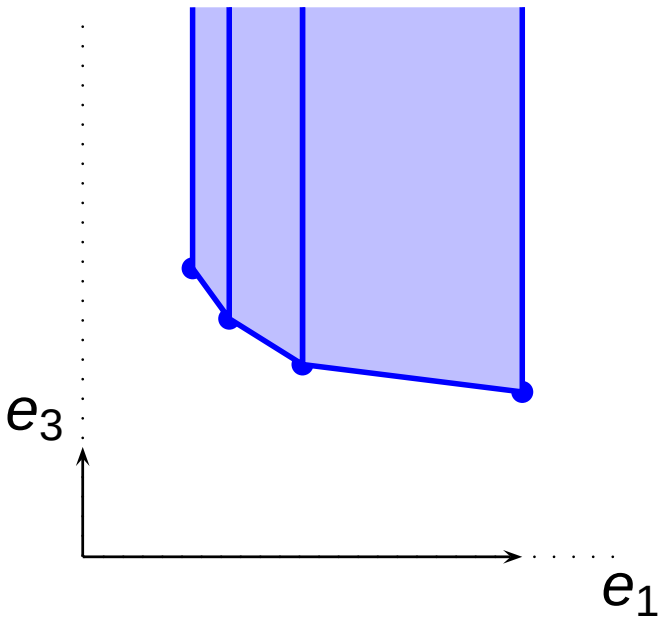


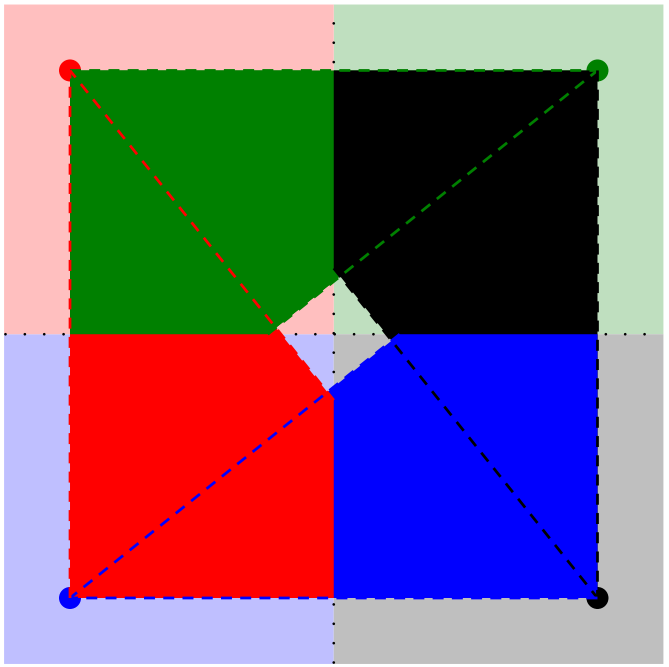


Remark

The MFP solution of this 3-dim cyclic IGTS is not polyhedral

Kleene iteration on 3-dim Cyclic Example

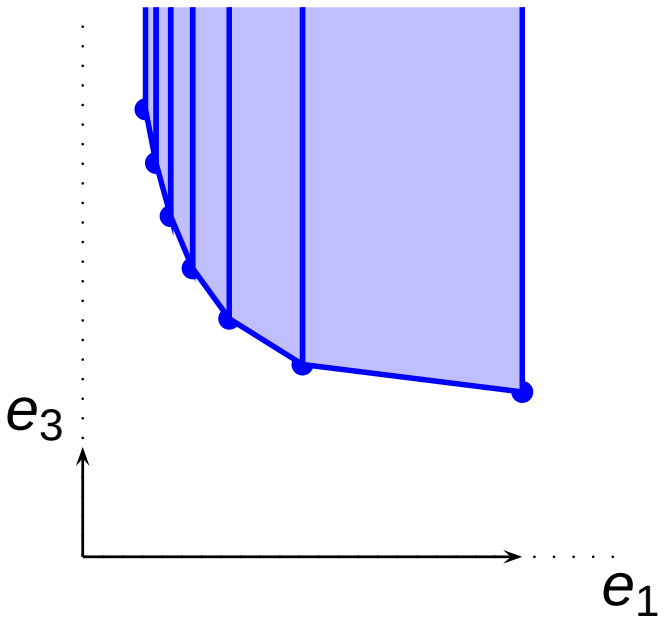


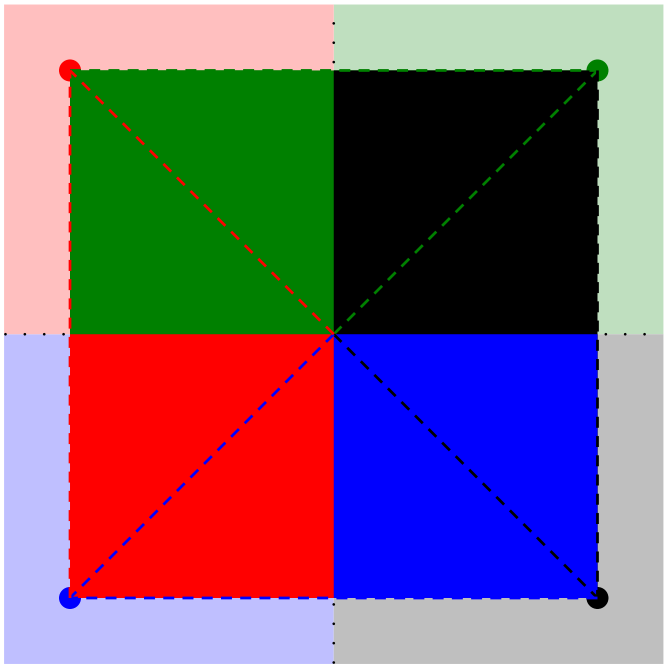


Remark

The MFP solution of this 3-dim cyclic IGTS is not polyhedral

Kleene iteration on 3-dim Cyclic Example

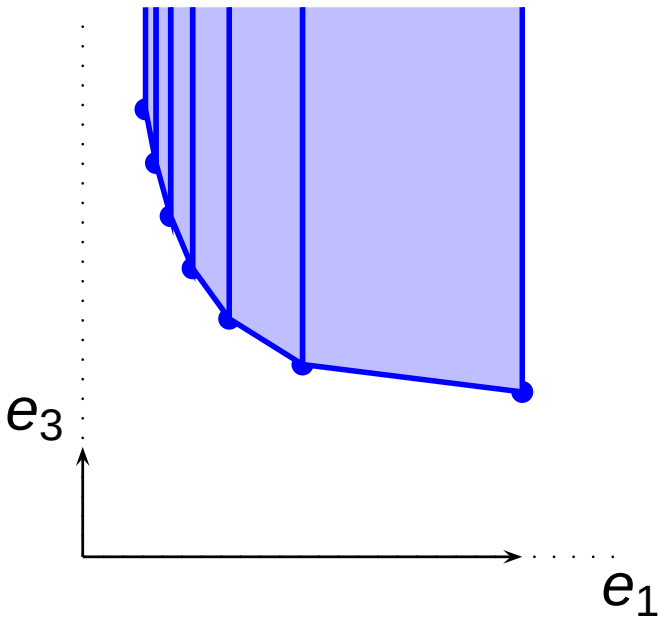


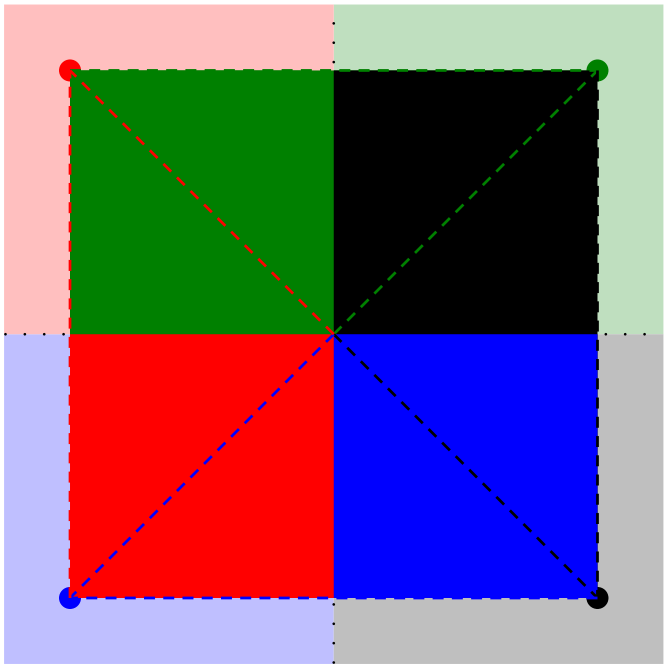


Remark

The MFP solution of this 3-dim cyclic IGTS is not polyhedral

Kleene iteration on 3-dim Cyclic Example





Remark

The MFP solution of this 3-dim cyclic IGTS is not polyhedral

Acceleration Results for Cycles

- 2-dim cyclic example with a real (non rational) polyhedral MFP solution
- 3-dim cyclic example with a non-polyhedral MFP solution

Question

Is the MFP polyhedral for all 2-dim cyclic IGTS?

Theorem

The MFP solution of any 2-dim IGTS is an algebraic polyhedron.

- An algebraic number is any real number definable in $\langle \mathbb{R}, +, \cdot, \leq \rangle$
- Algebraic polyhedrality is required even for cyclic 2-dim IGTS

Acceleration Results for Cycles

- 2-dim cyclic example with a real (non rational) polyhedral MFP solution
- 3-dim cyclic example with a non-polyhedral MFP solution

Question

Is the MFP polyhedral for all 2-dim cyclic IGTS?

Theorem

The MFP solution of any 2-dim IGTS is an algebraic polyhedron.

- An algebraic number is any real number definable in $\langle \mathbb{R}, +, \cdot, \leq \rangle$
- Algebraic polyhedrality is required even for cyclic 2-dim IGTS

Acceleration Results for Cycles

- 2-dim cyclic example with a real (non rational) polyhedral MFP solution
- 3-dim cyclic example with a non-polyhedral MFP solution

Question

Is the MFP polyhedral for all 2-dim cyclic IGTS?

Theorem

The MFP solution of any 2-dim IGTS is an algebraic polyhedron.

- An algebraic number is any real number definable in $\langle \mathbb{R}, +, \cdot, \leq \rangle$
- Algebraic polyhedrality is required even for cyclic 2-dim IGTS

Proof (1)

MFP Solution Expression

$$\mathrm{MFP}(X) = \bigsqcup_{\substack{x_0 \in \mathcal{X} \\ t_1 \cdots t_k \in L_{x_0,X}}} [\![t_k]\!] \circ \cdots \circ [\![t_1]\!](\Delta(X_0)) + \boxed{0^+ \mathrm{MFP}(X)}$$

where:

$$\Delta(X) = \rho_0(X) \sqcup \bigsqcup_{X \xrightarrow{G,\vec{d}} X'} \boxed{\mathrm{bd}(G) \cap \mathrm{MFP}(X)}$$

- $\mathrm{bd}(G)$ is the topological boundary of G
- $L_{X_0,X}$ is the set of simple paths from X_0 to X
- $0^+C = \{\vec{d} \mid C + \mathbb{R}_+\vec{d} \subseteq C\}$

Proof (2)

Observe that $0^+\,\mathrm{MFP}(X)$ is a *cone* in dimension 2.

$0^+\,\mathrm{MFP}(X)$

There exists $\vec{d_1}, \vec{d_2}, \vec{d_3} \in \mathbb{R}^2$ such that:
$0^+\,\mathrm{MFP}(X) = \mathbb{R}_+\vec{d_1} + \mathbb{R}_+\vec{d_1} + \mathbb{R}_+\vec{d_1}$

Reduce to the case G is an half-space.
$\Longrightarrow$ $\mathrm{bd}\,(G)$ is a *line*.

$\mathrm{bd}\,(G) \cap \mathrm{MFP}(X)$

There exists two half-spaces H_1, H_2 such that:
$\mathrm{bd}\,(G) \cap \mathrm{MFP}(X) = \mathrm{bd}\,(G) \cap H_1 \cap H_2$

Therefore the MFP solution is definable by a formula in $\langle \mathbb{R}, +, \cdot, \leq \rangle$.

Outline

Outline

Intervals of $\mathbb{Z}^n$

Complete Lattice $(\mathcal{Z}, \leq)$

Let $\mathcal{Z} = \mathbb{Z} \cup \{-\infty, +\infty\}$ with natural partial order $\leq$ defined by:

$$-\infty < \cdots < -2 < -1 < 0 < 1 < 2 < \cdots < +\infty$$

- greatest lower bound $\wedge$ satisfies: $a \wedge b = min(a, b)$ and $\bigwedge \emptyset = +\infty$
- least upper bound $\vee$ satisfies: $a \vee b = max(a, b)$ and $\bigvee \emptyset = -\infty$

Complete Lattice $(\mathcal{I}, \sqsubseteq)$

Set of all intervals $I = \{x \in \mathbb{Z} \mid a \leq x \leq b\}$ where $a, b \in \mathcal{Z}$, partially ordered by set inclusion

- greatest lower bound $\sqcap$ is set intersection $\cap$
- least upper bound $\sqcup$ is set union followed by "discrete convex hull"

Interval Constraint Systems

We consider data-flow programs over $(\mathcal{I}, \sqsubseteq)$ with transitions of the form:

$$X := I \qquad X := X_1 + X_2 \qquad X := X_1 \sqcap I$$
$$X := -X_1 \qquad X := X_1 \sqcup X_2$$
$$X := X_1 \cdot X_2$$

Allowed transfer functions

- constants
- full addition and subtraction
- full multiplication
- intersection with constants

We focus on the MFP solution, equivalently to the least solution of the constraint system where := is replaced with $\sqsupseteq$

Interval Constraints to Integer Constraints

Represent intervals by pairs of integers in $\mathcal{Z}$

$$I \mapsto (I^-, I^+) \qquad \text{where} \qquad \begin{cases} I^+ = \bigvee I \\ I^- = \bigvee(-I) = -\bigwedge I \end{cases}$$

Translations

$$X \sqsupseteq X_1 \quad \Leftrightarrow \quad \begin{cases} X^+ \geq X_1^+ \\ X^- \geq X_1^- \end{cases}$$

$$X \sqsupseteq -X_1 \quad \Leftrightarrow \quad \begin{cases} X^+ \geq X_1^- \\ X^- \geq X_1^+ \end{cases}$$

$$X \sqsupseteq X_1 + X_2 \quad \Leftrightarrow \quad \begin{cases} X^+ \geq X_1^+ + X_2^+ \\ X^- \geq X_1^- + X_2^- \end{cases}$$

But

$$X \sqsupseteq X_1 \sqcap I$$

is not equivalent to

$$\begin{cases} X^+ \geq X_1^+ \wedge I^+ \\ X^- \geq X_1^- \wedge I^- \end{cases}$$

Because $\rho(X_1) \sqcap I$ might be empty !

Interval Constraints to Integer Constraints

Represent intervals by pairs of integers in $\mathcal{Z}$

$$I \mapsto (I^-, I^+) \qquad \text{where} \qquad \begin{cases} I^+ = \bigvee I \\ I^- = \bigvee(-I) = -\bigwedge I \end{cases}$$

Translations

$$X \sqsupseteq X_1 \quad \Leftrightarrow \quad \begin{cases} X^+ \geq X_1^+ \\ X^- \geq X_1^- \end{cases}$$

$$X \sqsupseteq -X_1 \quad \Leftrightarrow \quad \begin{cases} X^+ \geq X_1^- \\ X^- \geq X_1^+ \end{cases}$$

$$X \sqsupseteq X_1 + X_2 \quad \Leftrightarrow \quad \begin{cases} X^+ \geq X_1^+ + X_2^+ \\ X^- \geq X_1^- + X_2^- \end{cases}$$

But

$$X \sqsupseteq X_1 \sqcap I$$

is not equivalent to

$$\begin{cases} X^+ \geq X_1^+ \wedge I^+ \\ X^- \geq X_1^- \wedge I^- \end{cases}$$

Because $\rho(X_1) \sqcap I$ might be empty !

The Test Functions

Definition

For every $v \in \mathcal{Z}$ and $\succ \in \{\geq, >\}$, define $\theta_{\succ v} : \mathcal{Z} \times \mathcal{Z} \to \mathcal{Z}$ by:

$$\theta_{\succ v}(z_1, z_2) = \begin{cases} z_2 & \text{if } z_1 \succ v \\ -\infty & \textit{otherwise} \end{cases}$$

- $I_1 \sqcap I_2 \neq \emptyset$ iff $I_1^- \geq -I_2^+$ and $I_1^+ \geq -I_2^-$
- if $I_1 \sqcap I_2 \neq \emptyset$ then $\begin{cases} (I_1 \sqcap I_2)^- = I_1^- \wedge I_2^- \\ (I_1 \sqcap I_2)^+ = I_1^+ \wedge I_2^+ \end{cases}$

Translation of $X \sqsubseteq X_1 \sqcap I$

$$X \sqsupseteq X_1 \sqcap I \Leftrightarrow \begin{cases} X^- \geq \theta_{\geq -I^+}(X_1^-, \theta_{\geq -I^-}(X_1^+, X_1^- \wedge I^-)) \\ X^+ \geq \theta_{\geq -I^+}(X_1^-, \theta_{\geq -I^-}(X_1^+, X_1^+ \wedge I^+)) \end{cases}$$

The Test Functions

Definition

For every $v \in \mathcal{Z}$ and $\succ \in \{\geq, >\}$, define $\theta_{\succ v} : \mathcal{Z} \times \mathcal{Z} \to \mathcal{Z}$ by:

$$\theta_{\succ v}(z_1, z_2) = \begin{cases} z_2 & \text{if } z_1 \succ v \\ -\infty & \textit{otherwise} \end{cases}$$

- $I_1 \sqcap I_2 \neq \emptyset$ iff $I_1^- \geq -I_2^+$ and $I_1^+ \geq -I_2^-$
- if $I_1 \sqcap I_2 \neq \emptyset$ then $\begin{cases} (I_1 \sqcap I_2)^- = I_1^- \wedge I_2^- \\ (I_1 \sqcap I_2)^+ = I_1^+ \wedge I_2^+ \end{cases}$

Translation of $X \sqsubseteq X_1 \sqcap I$

$$X \sqsupseteq X_1 \sqcap I \Leftrightarrow \begin{cases} X^- \geq \theta_{\geq -I^+}(X_1^-, \theta_{\geq -I^-}(X_1^+, X_1^- \wedge I^-)) \\ X^+ \geq \theta_{\geq -I^+}(X_1^-, \theta_{\geq -I^-}(X_1^+, X_1^+ \wedge I^+)) \end{cases}$$

The Test Functions

Definition

For every $v \in \mathcal{Z}$ and $\succ \in \{\geq, >\}$, define $\theta_{\succ v} : \mathcal{Z} \times \mathcal{Z} \to \mathcal{Z}$ by:

$$\theta_{\succ v}(z_1, z_2) = \begin{cases} z_2 & \text{if } z_1 \succ v \\ -\infty & \textit{otherwise} \end{cases}$$

- $I_1 \sqcap I_2 \neq \emptyset$ iff $I_1^- \geq -I_2^+$ and $I_1^+ \geq -I_2^-$
- if $I_1 \sqcap I_2 \neq \emptyset$ then $\begin{cases} (I_1 \sqcap I_2)^- = I_1^- \wedge I_2^- \\ (I_1 \sqcap I_2)^+ = I_1^+ \wedge I_2^+ \end{cases}$

Translation of $X \sqsubseteq X_1 \sqcap I$

$$X \sqsupseteq X_1 \sqcap I \Leftrightarrow \begin{cases} X^- \geq \theta_{\geq -I^+}(X_1^-, \theta_{\geq -I^-}(X_1^+, X_1^- \wedge I^-)) \\ X^+ \geq \theta_{\geq -I^+}(X_1^-, \theta_{\geq -I^-}(X_1^+, X_1^+ \wedge I^+)) \end{cases}$$

The Multiplication Functions

Definition

Define the multiplication functions $\mathrm{mul}_+, \mathrm{mul}_- : \mathcal{Z} \times \mathcal{Z} \to \mathcal{Z}$ by:

$$\mathrm{mul}_+(z_1, z_2) = \begin{cases} z_1 \cdot z_2 & \text{if } z_1, z_2 > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{mul}_-(z_1, z_2) = \begin{cases} -z_1 \cdot z_2 & \text{if } z_1, z_2 < 0 \\ 0 & \text{otherwise} \end{cases}$$

Translation of $X \sqsubseteq X_1 \cdot X_2$

When $\rho(X_1) \subseteq \mathbb{N}$ and $\rho(X_2) \subseteq \mathbb{N}$ then $X \sqsupseteq X_1 \cdot X_2$ is equivalent to:

$$\begin{cases} X^- \geq \theta_{>-\infty}(X_1^-, \theta_{>-\infty}(X_1^+, \theta_{>-\infty}(X_2^-, \theta_{>-\infty}(X_2^+, \mathrm{mul}_-(X_1^-, X_2^-))))) \\ X^+ \geq \theta_{>-\infty}(X_1^+, \theta_{>-\infty}(X_1^-, \theta_{>-\infty}(X_2^+, \theta_{>-\infty}(X_2^-, \mathrm{mul}_+(X_1^+, X_2^+))))) \end{cases}$$

The Multiplication Functions

Definition

Define the multiplication functions $\mathrm{mul}_+, \mathrm{mul}_- : \mathcal{Z} \times \mathcal{Z} \to \mathcal{Z}$ by:

$$\mathrm{mul}_+(z_1, z_2) = \begin{cases} z_1 \cdot z_2 & \text{if } z_1, z_2 > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{mul}_-(z_1, z_2) = \begin{cases} -z_1 \cdot z_2 & \text{if } z_1, z_2 < 0 \\ 0 & \text{otherwise} \end{cases}$$

Translation of $X \sqsubseteq X_1 \cdot X_2$

When $\rho(X_1) \subseteq \mathbb{N}$ and $\rho(X_2) \subseteq \mathbb{N}$ then $X \sqsupseteq X_1 \cdot X_2$ is equivalent to:

$$\begin{cases} X^- \geq \theta_{>-\infty}(X_1^-, \theta_{>-\infty}(X_1^+, \theta_{>-\infty}(X_2^-, \theta_{>-\infty}(X_2^+, \mathrm{mul}_-(X_1^-, X_2^-))))) \\ X^+ \geq \theta_{>-\infty}(X_1^+, \theta_{>-\infty}(X_1^-, \theta_{>-\infty}(X_2^+, \theta_{>-\infty}(X_2^-, \mathrm{mul}_+(X_1^+, X_2^+))))) \end{cases}$$

Translation into Positive Multiplication Form

Replace each constraint $X \supseteq X_1 \cdot X_2$ by:

$$
\begin{array}{lll}
X \supseteq X_{1,u} \cdot X_{2,u} & X_{1,u} \supseteq X_1 \sqcap \mathbb{N} & \\
X \supseteq X_{1,l} \cdot X_{2,l} & X_{2,u} \supseteq X_2 \sqcap \mathbb{N} & \\
X \supseteq -X' & & \\
X' \supseteq X_{1,u} \cdot X_{2,l} & X_{1,l} \supseteq X'_1 \sqcap \mathbb{N} & X'_1 \supseteq -X_1 \\
X' \supseteq X_{1,l} \cdot X_{2,u} & X_{2,l} \supseteq X'_2 \sqcap \mathbb{N} & X'_2 \supseteq -X_2
\end{array}
$$

- $X_{i,u}$ corresponds to the "positive part" of X_i
- $X_{i,l}$ corresponds to the "negative part" of X_i

Property of Transformed Contraint System

The least solution ρ satisfies $\rho(X_1) \subseteq \mathbb{N}$ and $\rho(X_2) \subseteq \mathbb{N}$ for any multiplicative constraint $X \supseteq X_1 \cdot X_2$

Outline

Bounded-Increasing Integer Constraint Systems

We have reduced interval contraint systems to constraint systems over $(\mathcal{Z}, \leq)$ with constraints of the form:

$$X \geq z \qquad \begin{array}{l} X \geq X_1 + X_2 \\ X \geq \mathsf{mul}_+(X_1, X_2) \\ X \geq \mathsf{mul}_-(X_1, X_2) \end{array} \qquad \begin{array}{l} X \geq X_1 \wedge z \\ X \geq \theta_{\geq z}(X_1, X_2) \\ X \geq \theta_{>z}(X_1, X_2) \end{array}$$

Definition

A monotonic function $f \in \mathcal{Z}^k \to \mathcal{Z}$ is bounded-increasing if $f(\vec{a}) < f(\vec{b})$ for every $\vec{a} < \vec{b}$ such that $f(\bot) < f(\vec{a})$ and $f(\vec{b}) < f(\top)$

Except for test functions, all of the above transfer functions are bounded-increasing

Bounded-Increasing Integer Constraint Systems

We have reduced interval contraint systems to constraint systems over $(\mathcal{Z}, \leq)$ with constraints of the form:

$$X \geq z \qquad X \geq X_1 + X_2 \qquad X \geq X_1 \wedge z$$
$$X \geq \mathrm{mul}_+(X_1, X_2) \qquad X \geq \theta_{\geq z}(X_1, X_2)$$
$$X \geq \mathrm{mul}_-(X_1, X_2) \qquad X \geq \theta_{> z}(X_1, X_2)$$

Definition

A monotonic function $f \in \mathcal{Z}^k \to \mathcal{Z}$ is bounded-increasing if $f(\vec{a}) < f(\vec{b})$ for every $\vec{a} < \vec{b}$ such that $f(\bot) < f(\vec{a})$ and $f(\vec{b}) < f(\top)$

Except for test functions, all of the above transfer functions are bounded-increasing

Computation of the Least Solution

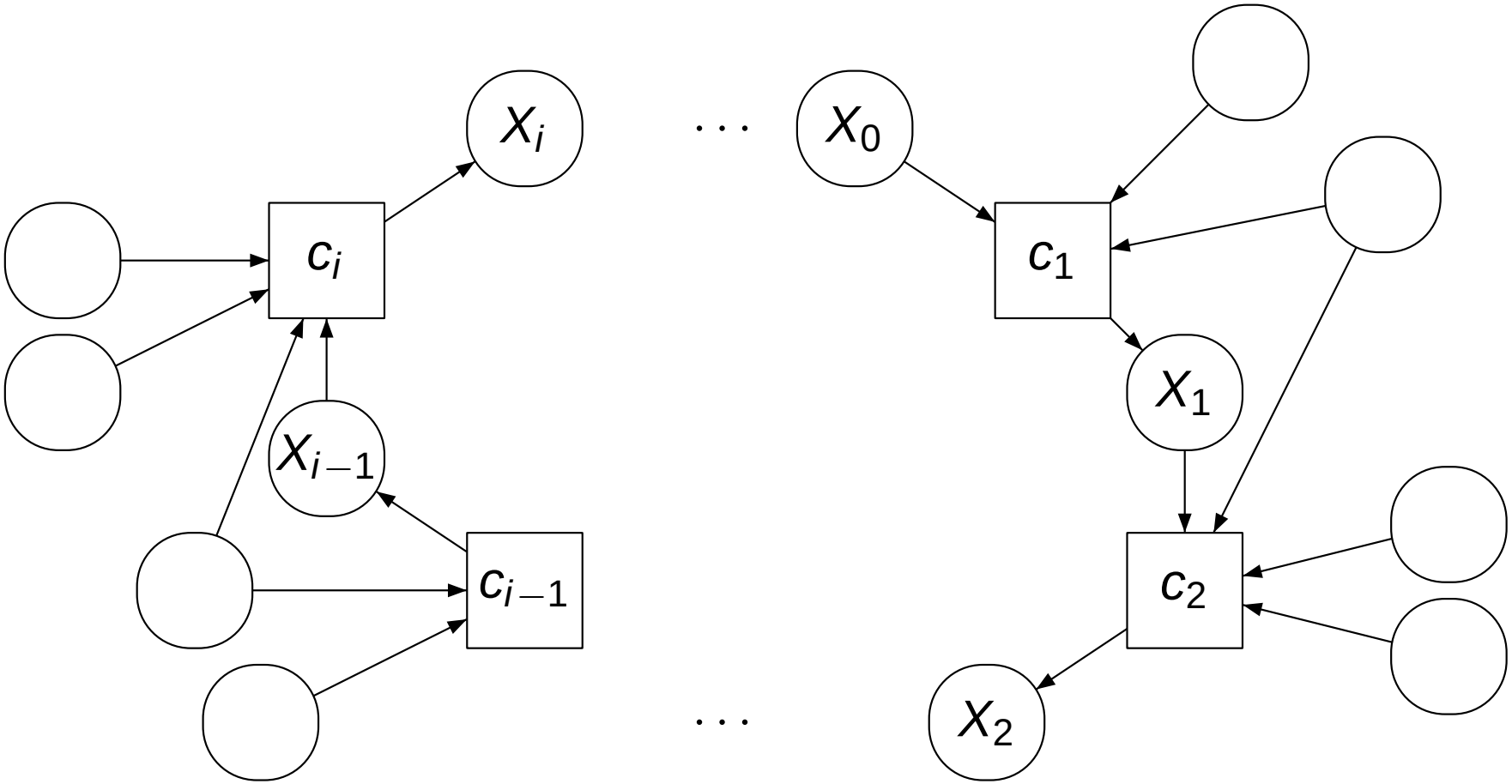
Definition

A constraint $X \geq f(X_1, \ldots, X_n)$ is called saturated by a valuation ρ when $\rho(X) \geq f(\top)$

Main Ideas of the Algorithm

- Iterative forward propagation
- Keep track for each variable of the last constraint that updated its value
- When a cycle of updates appears, accelerate it to saturate at least one constraint
- Inject test constraints only once they become “active”

Cyclic Constraint Systems



Algorithm for the Cyclic Bounded-Increasing Case

1 CyclicSolve($\mathcal{S} = (\mathcal{X}, C)$: cyclic bounded-increasing constraint system,
2 ρ_0 : valuation)
3 let $X_0 \to c_1 \to X_1 \cdots \to c_n \to X_n = X_0$ be the "unique" elementary cycle
4 $\rho \leftarrow \rho_0$
5 for $i = 1$ to n do
6 $\rho \leftarrow \rho \vee [\![c_i]\!](\rho)$
7 for $i = 1$ to n do
8 $\rho \leftarrow \rho \vee [\![c_i]\!](\rho)$
9 if $\rho \geq [\![C]\!](\rho)$
10 return ρ
11 for $i = 1$ to n do
12 $\rho(X_i) \leftarrow +\infty$
13 for $i = 1$ to n do
14 $\rho \leftarrow \rho \wedge [\![c_i]\!](\rho)$
15 for $i = 1$ to n do
16 $\rho \leftarrow \rho \wedge [\![c_i]\!](\rho)$
17 return ρ

Algorithm for the General Bounded-Increasing Case

1 SolveBI($\mathcal{S} = (\mathcal{X}, C)$: bounded-increasing constraint system,
2 ρ_0 : valuation)
3 $\rho \leftarrow \rho_0 \vee [\![C]\!](\rho_0)$
4 `while` $[\![C]\!](\rho) \not\sqsubseteq \rho$
5 $\lambda \leftarrow \emptyset$ $\{$ λ is a partial function from $\mathcal{X}$ to C $\}$
6 `repeat` $|C| + 1$ times
7 `for each` $c \in C$
8 `if` $\rho \not\geq [\![c]\!](\rho)$
9 $\rho \leftarrow \rho \vee [\![c]\!](\rho)$
10 $\lambda(X) \leftarrow c$, where X is the input variable of c
11 `if` there is a cycle $X_0 \rightarrow \lambda(X_1) \rightarrow X_1 \cdots \lambda(X_n) \rightarrow X_0$
12 $\mathcal{S}' \leftarrow (\mathcal{X}, \{\lambda(X_1), \ldots, \lambda(X_n)\})$
13 $\rho' \leftarrow$ CyclicSolve$(\mathcal{S}', \rho)$
14 $\rho \leftarrow \rho \vee \rho'$
15 `return` ρ

Algorithm for General Integer Constraint Systems

Active Test Constraints

- A test constraint $c = X \geq \theta_{\succ z}(X_1, X_2)$ is active for ρ if $\rho(X_1) \succ z$
- Its active form act(c) is the constraint $X \geq X_2$

```
1  SolveInteger(S = (X, C) : integer constraint system)
2  ρ ← ⊥
3  C_t ← set of test constraints in C
4  C' ← set of bounded-increasing constraints in C
5  while [[C]](ρ) ⋢ ρ
6        ρ ← SolveBI((X, C'), ρ)
7        for each c ∈ C_t
8              if c is active for ρ
9                    C_t ← C_t \ {c}
10                   C' ← C' ∪ {act(c)}
11 return ρ
```

Correctness and Complexity Results

Size $|\mathcal{S}|$ of a constraint system $\mathcal{S} = (\mathcal{X}, C)$ defined by $|\mathcal{S}| = |\mathcal{X}| + |C|$

Theorem

The algorithm SolveInteger computes the least solution of a system of (test and bounded-increasing) integer constraints $\mathcal{S}$ by performing $O(|\mathcal{S}|^3)$ integer comparisons and image computations by bounded-increasing transfer functions of $\mathcal{S}$

Theorem

The least solution of an interval constraint system $\mathcal{S}$ can by computed in time $O(|\mathcal{S}|^3)$ with integer operations performed in $O(1)$

Outline

Summary (1)

Acceleration Framework for Data-Flow Analysis

- Generalizes “standard” acceleration principles from concrete to abstract data-flow analysis
- Tradeoff between reachability set computation and data-flow analysis with widenings / narrowings

Application of Framework

- Convex data-flow analysis
  - computation of the MOP and MFP solution for cyclic GTS
  - better acceleration strategy than previous work for self-loops
- Interval Constraint Solving
  - interval constraints with full multiplication (but restricted $\sqcap$)
  - instanciation of the generic AcceleratedMFP semi-algorithm
  - efficient approach: cubic-time complexity, on-the-fly

Summary (1)

Acceleration Framework for Data-Flow Analysis

- Generalizes “standard” acceleration principles from concrete to abstract data-flow analysis
- Tradeoff between reachability set computation and data-flow analysis with widenings / narrowings

Application of Framework

- Convex data-flow analysis
  - computation of the MOP and MFP solution for cyclic GTS
  - better acceleration strategy than previous work for self-loops
- Interval Constraint Solving
  - interval constraints with full multiplication (but restricted $\sqcap$)
  - instanciation of the generic AcceleratedMFP semi-algorithm
  - efficient approach: cubic-time complexity, on-the-fly

Summary (2)

		Guarded Translation Systems		
		Self-loops	Cyclic	General
MOP	$n \geq 1$	Rational Poly.	Rational Poly.	Not Polyhedral
MFP	1	Rational Poly.	Rational Poly.	Rational Poly.
	2	Rational Poly.	Algebraic Poly.	Algebraic Poly.
	$n \geq 3$	Rational Poly.	Not Polyhedral	Not Polyhedral

- Polyhedra are computable for *Rational Poly.* and *Algebraic Poly.*
- Results on self-loops carry over to singly initialized cycles

Related Work & Future Work

Related Work

- Interval analysis [Su & Wagner, TACAS'04], [Seidl & Gawlitza, ESOP'07]
  - No polynomial-time algorithm for constraints with full multiplication
- Abstract acceleration for convex polyhedra [Gonnord & Halbwachs, SAS'06]
  - Acceleration technique for two self-loops, operations include reset
  - Incomplete for single self-loops

Future Work

- Multiple self-loops
- Other abstract lattices
  - octogons [Miné, AST'01]
  - templates [Sankaranarayanan *et al.*, VMCAI'05]
  - two variables per linear inequality [Simon *et al.*, LOPSTR'02]

Related Work & Future Work

Related Work

- Interval analysis [Su & Wagner, TACAS'04], [Seidl & Gawlitza, ESOP'07]
  - No polynomial-time algorithm for constraints with full multiplication
- Abstract acceleration for convex polyhedra [Gonnord & Halbwachs, SAS'06]
  - Acceleration technique for two self-loops, operations include reset
  - Incomplete for single self-loops

Future Work

- Multiple self-loops
- Other abstract lattices
  - octogons [Miné, AST'01]
  - templates [Sankaranarayanan *et al.*, VMCAI'05]
  - two variables per linear inequality [Simon *et al.*, LOPSTR'02]