A Gentle Introduction to Static Analysis (2)

Woosuk Lee

CSE 6049 Program Analysis

Hanyang University, Korea

Some slides are borrowed from http://ropas.snu.ac.kr/~kwang/4541.664A/21/0-overview.pdf

Static Analysis

A general method for
automatic and sound approximation of
sw run-time behaviors
before the execution

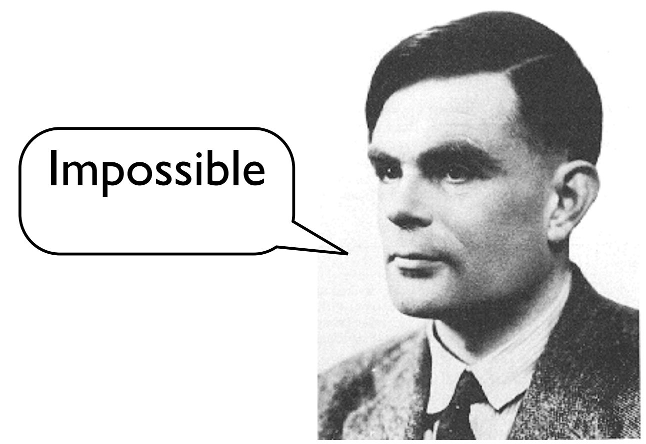
- "before": statically, without running sw
- "automatic": sw analyzes sw
- "sound": all possibilities into account
- "approximation": cannot be exact
- "general": for any source language and property
  - C, C++, C#, F#, Java, JavaScript, ML, Scala, Python, JVM, Dalvik, x86, Excel, etc
  - "buffer-overrun?", "memory leak?", "type errors?", "x = y at line 2?", "memory use $\leq \boldsymbol{2K}$?", etc

<u>Abstract</u> Interpretation

- A powerful framework for designing correct static analysis
  - "**framework**" : correct static analysis comes out, reusable
  - "**powerful**" : all static analyses are understood in this framework
  - "**simple**" : prescription is simple
  - "**eye-opening**" : any static analysis is an abstract interpretation

Why Abstraction?

- Without abstraction,
  - can’t capture all possible executions
  - can’t terminate



Alan Turing

- **<u>Abstraction ≠ omission</u>**
  - reality: {2, 4, 6, 8, … }
  - “even number” (abstraction) vs “multiple of 4” (omission)

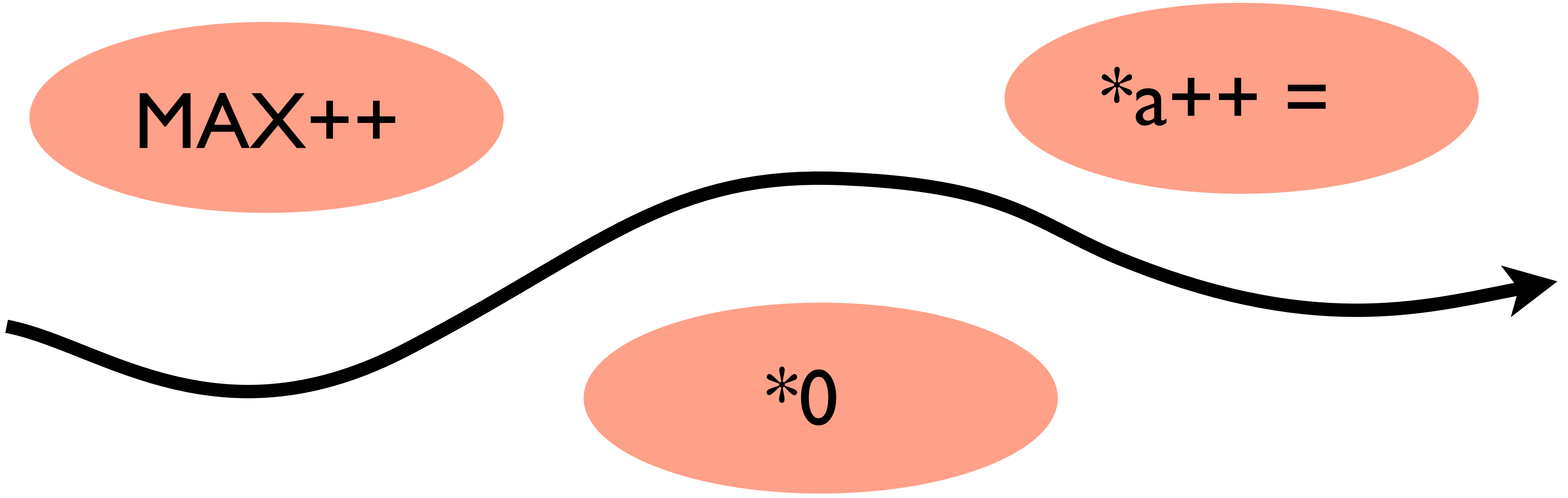
Example

- Q: What are the possible output values?

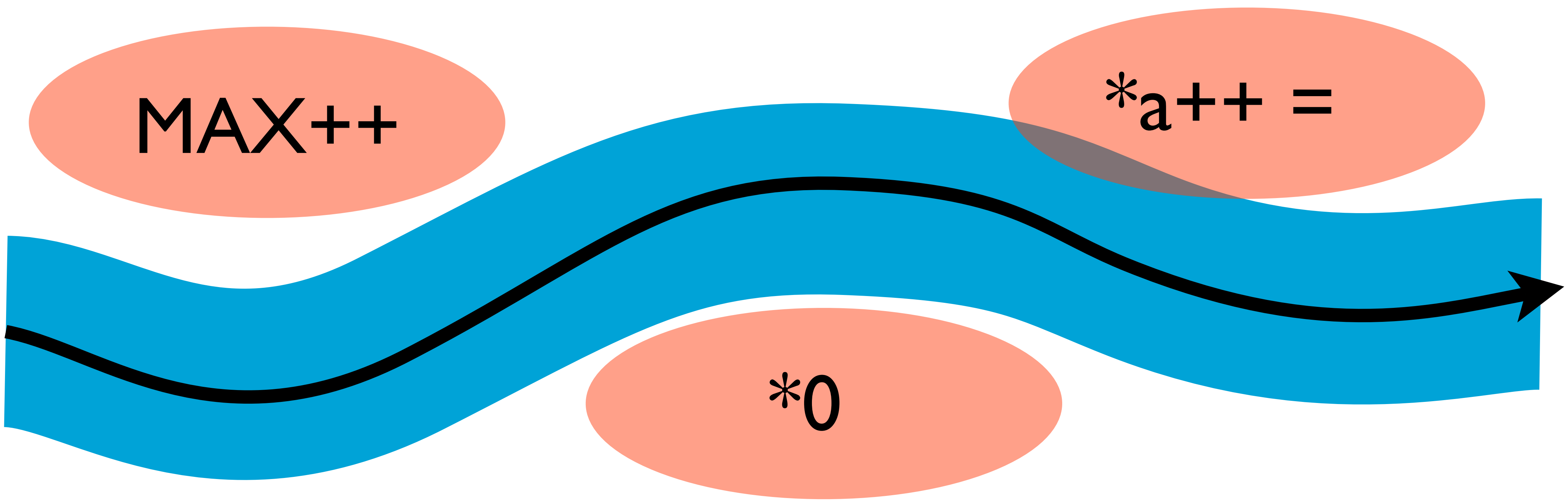
```
x = 3;
while (*) {
  x += 2;
}
x -= 1;
print(x);
```

- Concrete interpretation: 2,4, … infinitely many possible values
- Abstract interpretation 1: "integers" (coarse)
- Abstract interpretation 2: "positive integers" (precise)
- Abstract interpretation 3: "positive even integers" (more precise)

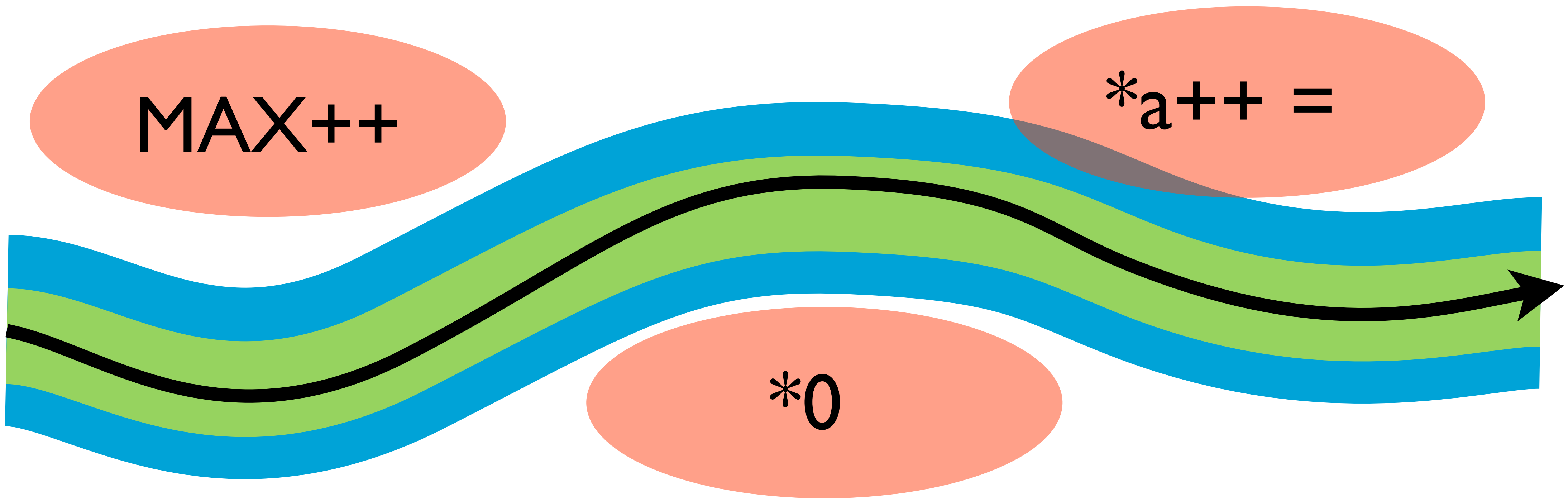
Abstraction



Abstraction



Abstraction



An Intuitive Explanation of Abstract Interpretation

Example Language

Initialization with a point that is non-deterministically chosen in a fixed region (e.g., [0,1] x [0,1] square)

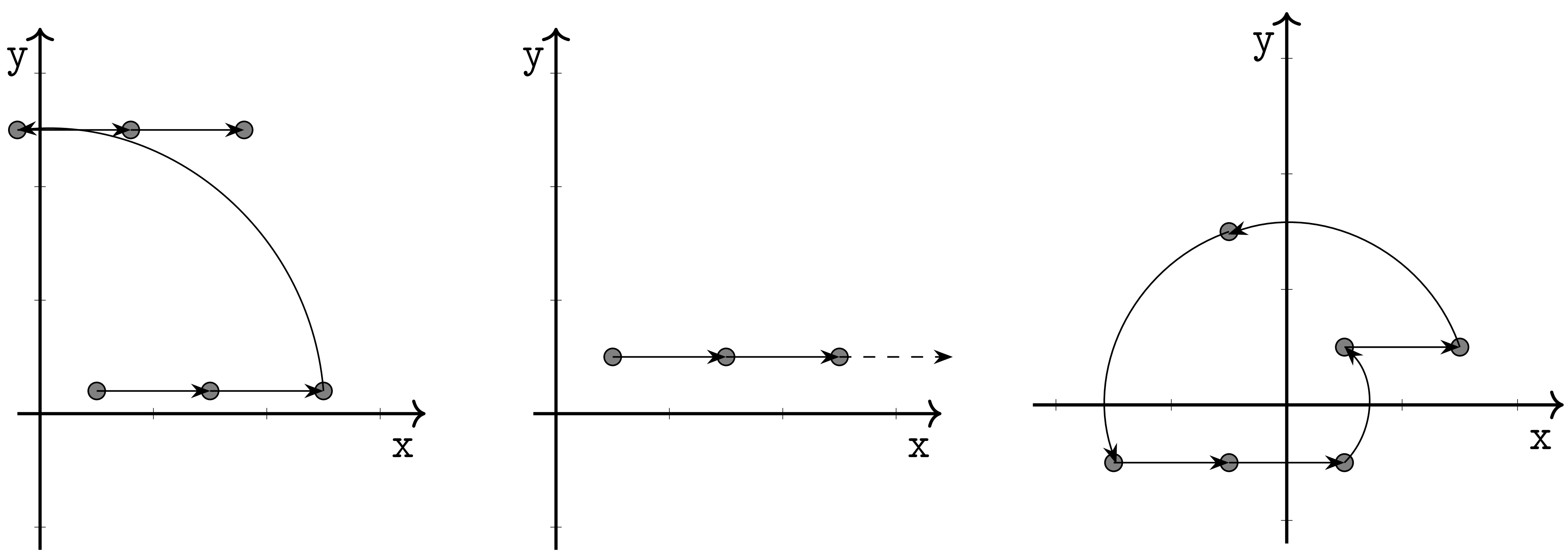
$$
\begin{array}{lcll}
\texttt{p} & ::= & \texttt{init}(\mathfrak{R}) & \text{initialization, with a state in } \mathfrak{R} \\
& | & \texttt{translation}(u, v) & \text{translation by vector } (u, v) \\
& | & \texttt{rotation}(u, v, \theta) & \text{rotation by center } (u, v) \text{ and angle } \theta \\
& | & \texttt{p} \; ; \; \texttt{p} & \text{sequence of operations} \\
& | & \{\texttt{p}\}\texttt{or}\{\texttt{p}\} & \text{non-deterministic choice} \\
& | & \texttt{iter}\{\texttt{p}\} & \text{non-deterministic iterations}
\end{array}
$$

All programs start with an initialization statement.

Semantics

Example (Semantics)

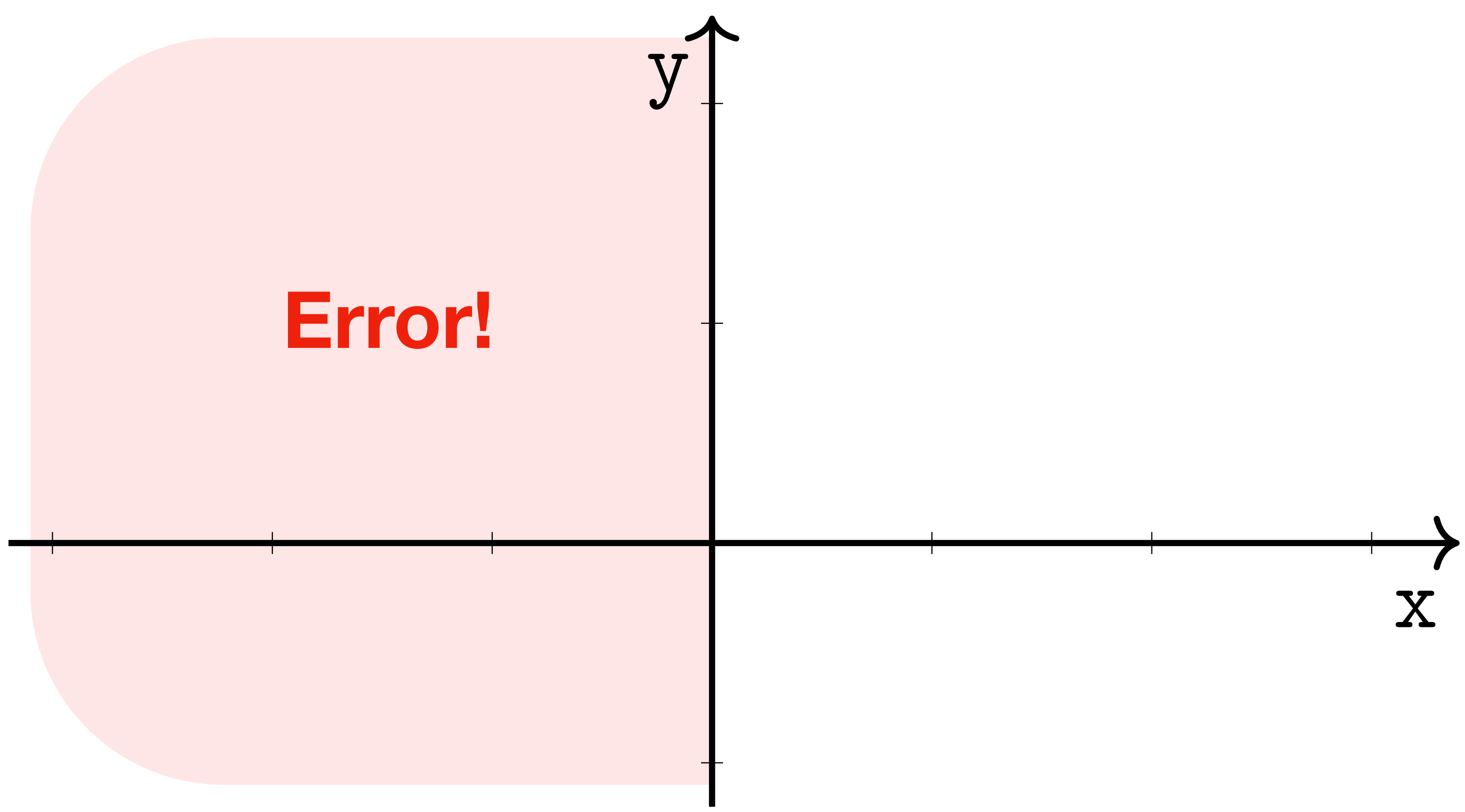
$\texttt{init}([0,1] \times [0,1]);$
$\texttt{translation}(1,0);$
$\texttt{iter}\{$
$\quad \{$
$\quad\quad \texttt{translation}(1,0)$
$\quad \}\texttt{or}\{$
$\quad\quad \texttt{rotation}(0,0,90^\circ)$
$\quad \}$
$\}$



Analysis Goal Is Safety Property: Reachability

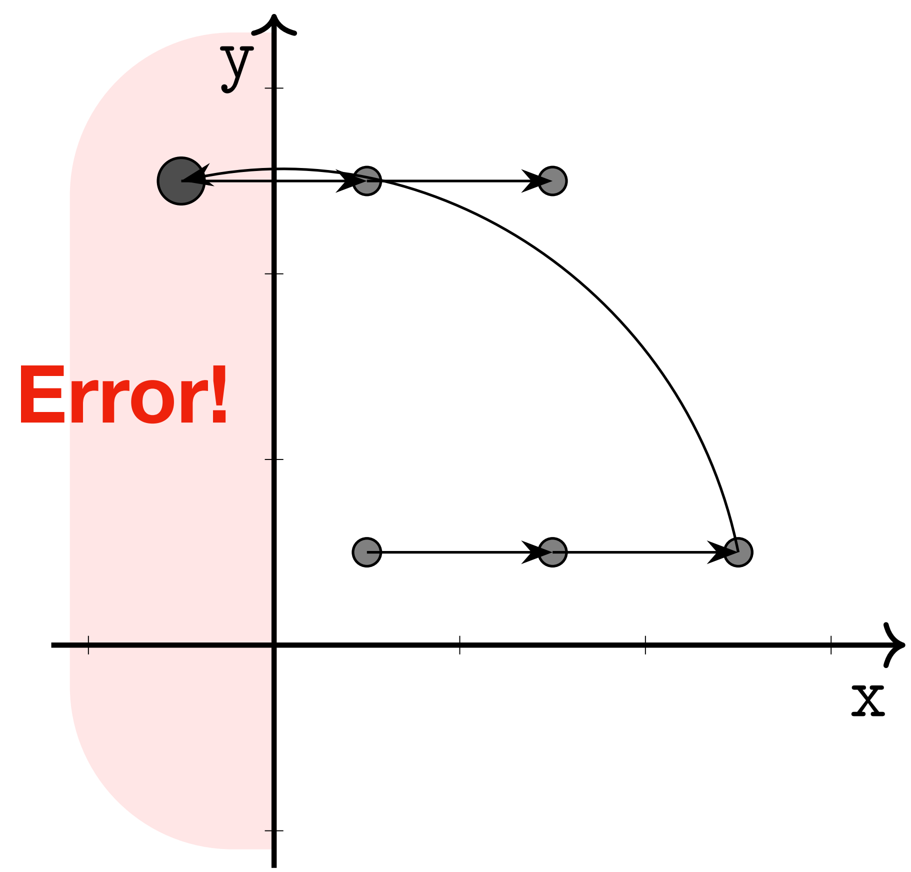
Analyze the set of reachable points, to check if the set intersects with a hypothetical error zone:

$$\mathcal{D} = \{(x, y) \mid x < 0\}$$



Correct / Incorrect Executions

- Our goal: prove $\neg\mathcal{D}$

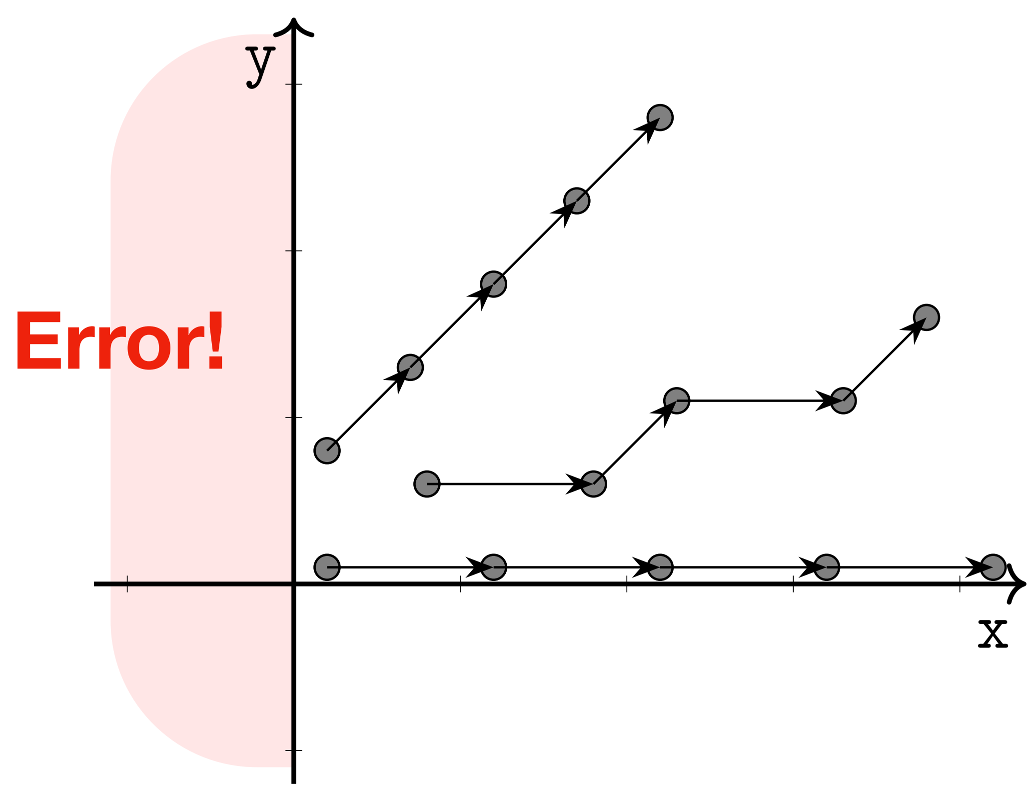


(a) An incorrect execution

An Example Safe Program

Example

$\texttt{init}([0,1] \times [0,1]);$
$\texttt{iter}\{$
$\quad\{$
$\quad\quad \texttt{translation}(1,0)$
$\quad\}\texttt{or}\{$
$\quad\quad \texttt{translation}(0.5,0.5)$
$\quad\}$
$\}$



Need for Static Analysis for Proving $\neg\mathcal{D}$

- How can we check $\neg\mathcal{D}$ for any given program?
- Enumeration of all executions does not work!
  - The set of possible initial states is infinite.
  - The length of executions may be infinite.
  - The set of possible series of non-deterministic choices is infinite.

How to Finitely Over-Approximate the Set of Reachable Points?

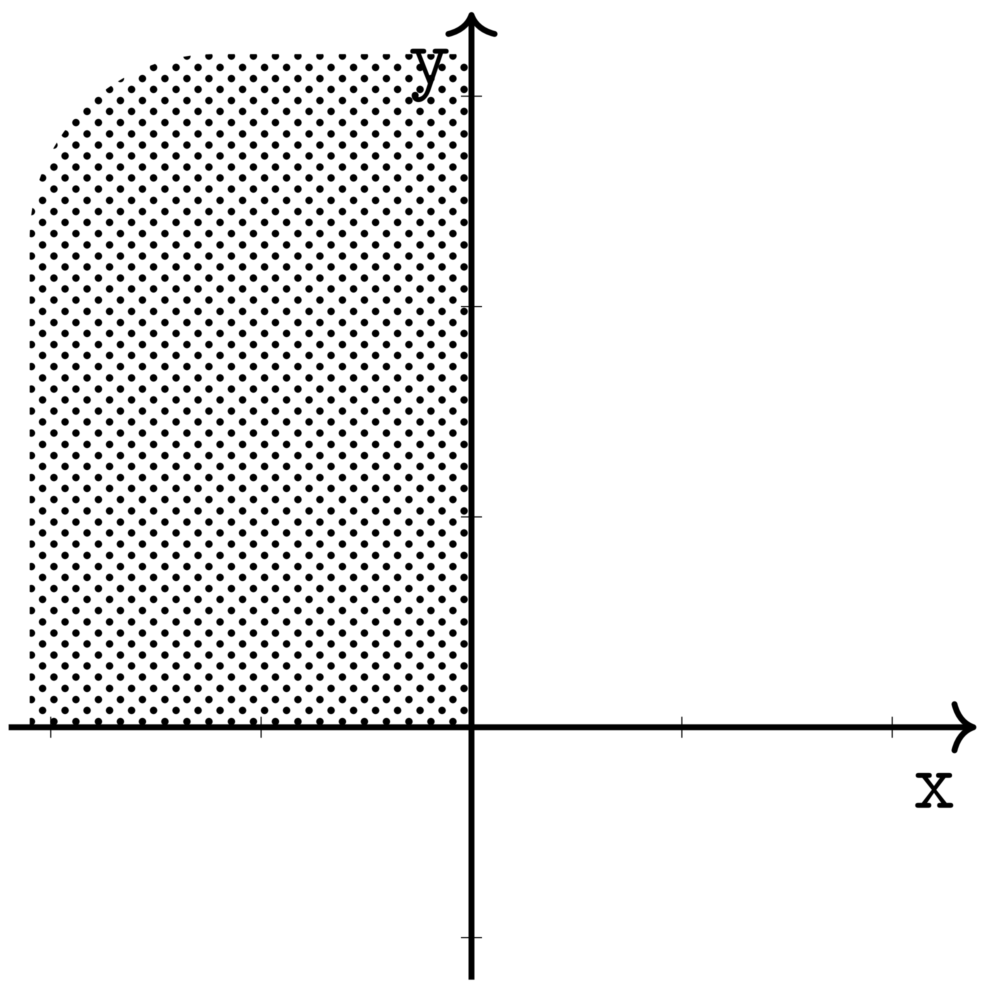
Definition (Abstraction)

We call *abstraction* a set $\mathcal{A}$ of logical properties of program states, which are called *abstract properties* or *abstract elements*. A set of abstract properties is called an *abstract domain*.

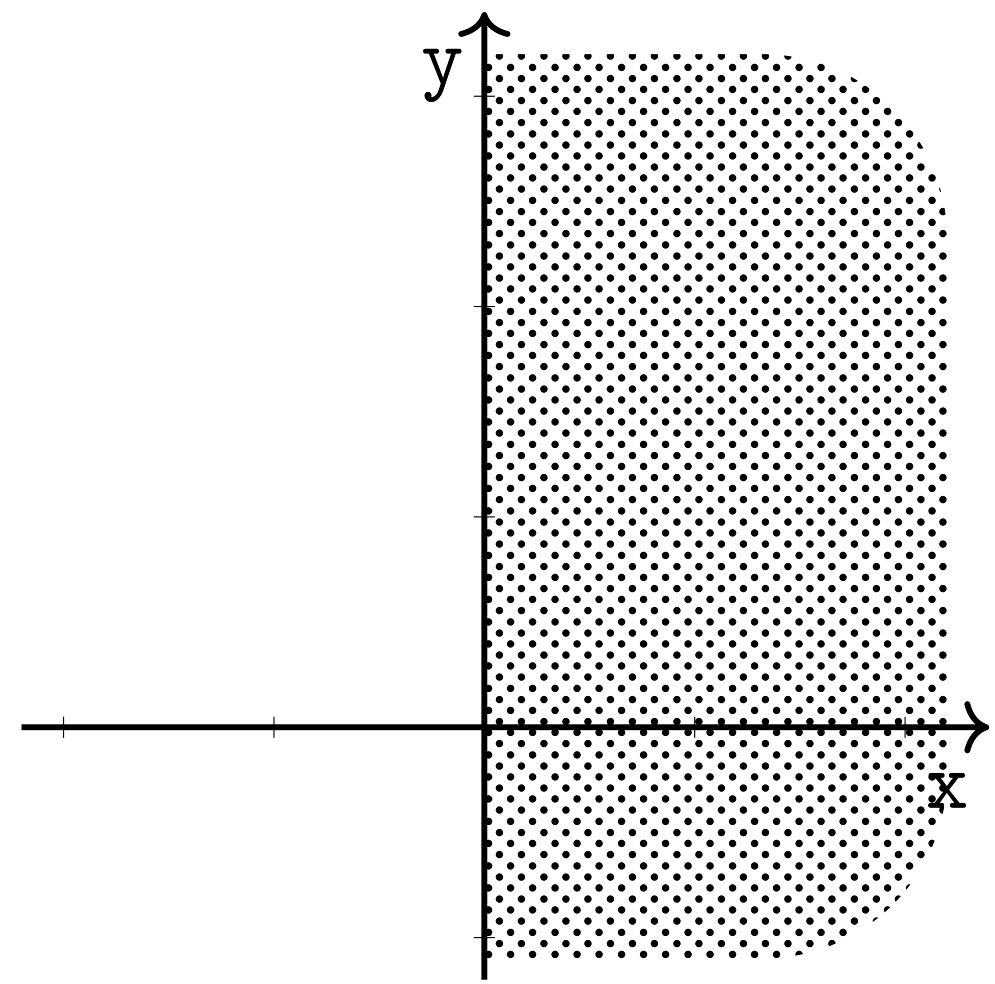
Definition (Concretization)

Given an abstract element a of $\mathcal{A}$, we call *concretization* the set of program states that satisfy it. We denote it by $\gamma(a)$.

Abstraction Example 1: Sign Abstraction



(c) Concretization of $[\mathtt{x} \leq 0, \mathtt{y} \geq 0]$

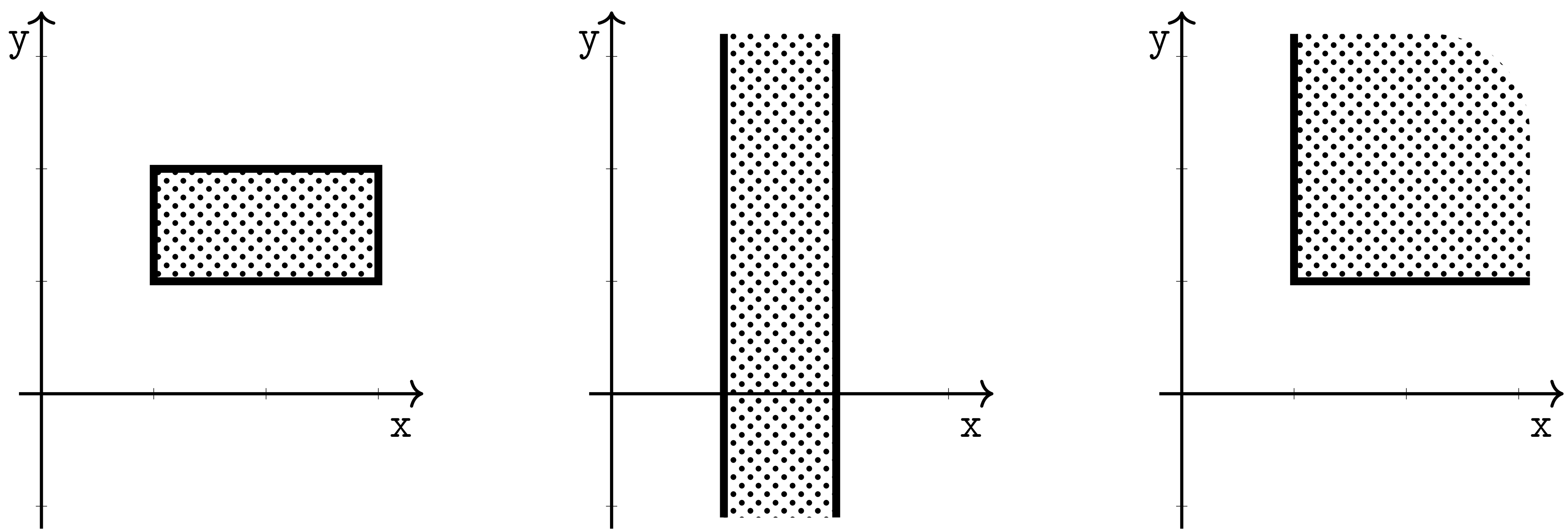


(d) Concretization of $[\mathtt{x} \geq 0]$

Figure: Signs abstraction

Abstraction Example 2: Interval Abstraction

The abstract elements: conjunctions of non-relational inequality constraints: $c_1 \le \mathtt{x} \le c_2$, $c_1' \le \mathtt{y} \le c_2'$



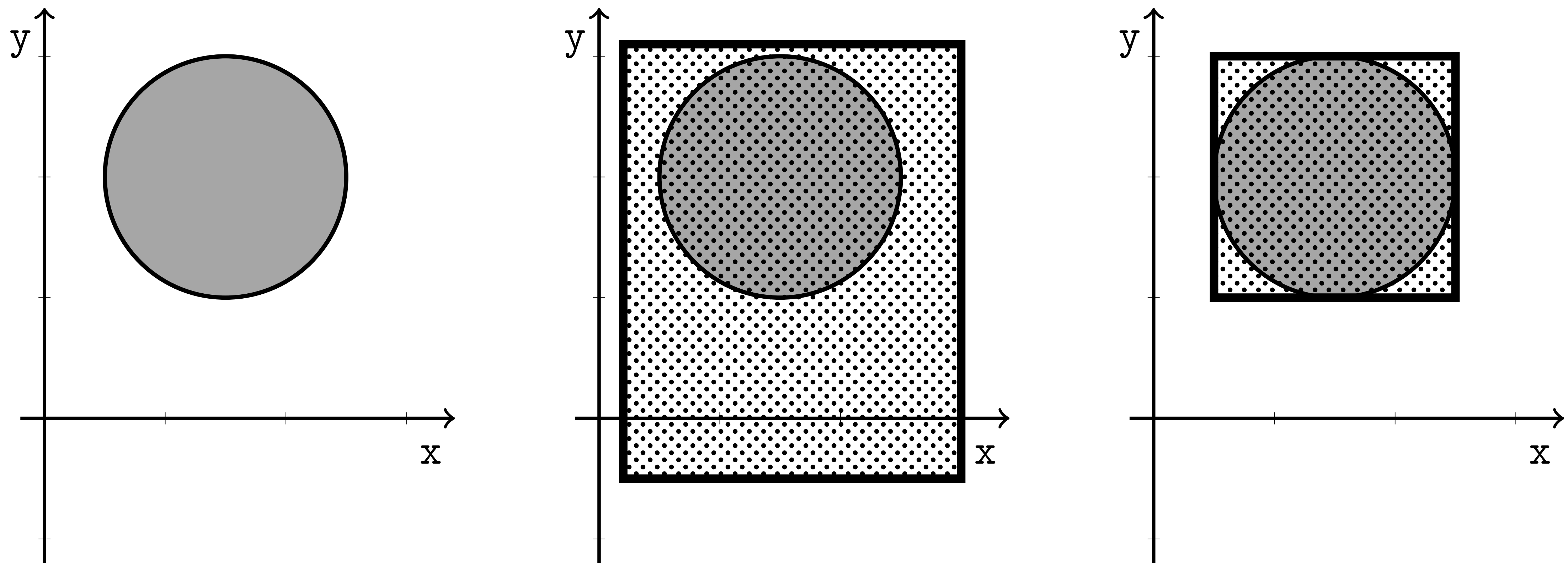
(a) Concretization of $[1 \le \mathtt{x} \le 3, 1 \le \mathtt{y} \le 2]$

(b) Concretization of $[1 \le \mathtt{x} \le 2]$

(c) Concretization of $[1 \le \mathtt{x}, 1 \le \mathtt{y}]$

Figure: Intervals abstraction

Best Abstraction



(a) A concrete set (b) Abstractions (c) Best abstraction

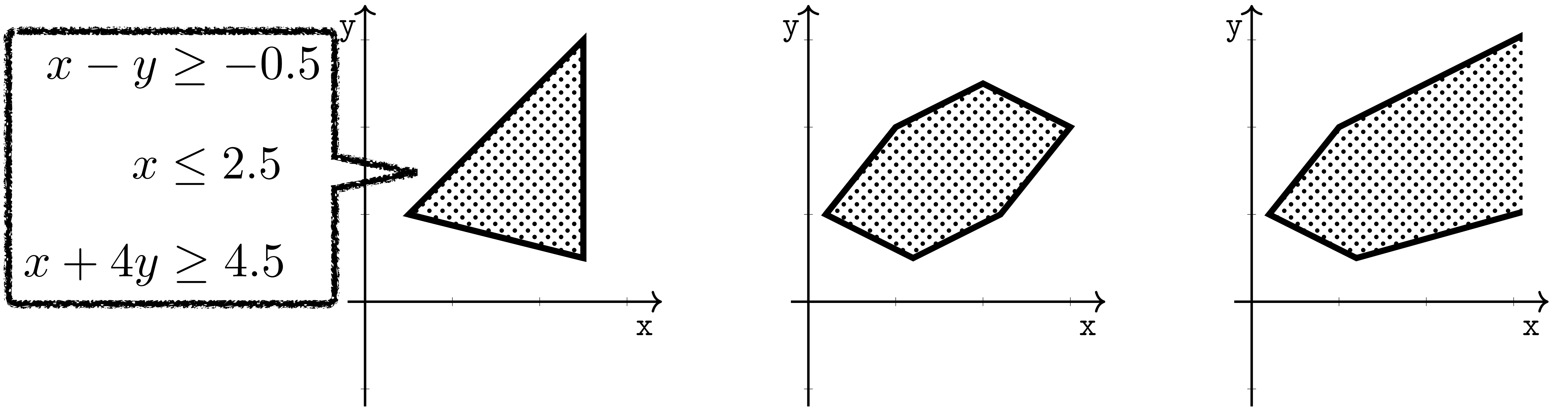
Figure 2.7
Best abstraction

Best Abstraction

- We say a is the best abstraction of the concrete set S iff
  - $S \subseteq \gamma(a)$, and
  - for any a' such that $S \subseteq \gamma(a')$, a' is a coarser abstraction than a.

Abstraction Example 3: Convex Polyhedra Abstraction

The abstract elements: conjunctions of linear inequality constraints: $c_1\mathrm{x} + c_2\mathrm{y} \leq c_3$



(a) Concretization of a_0

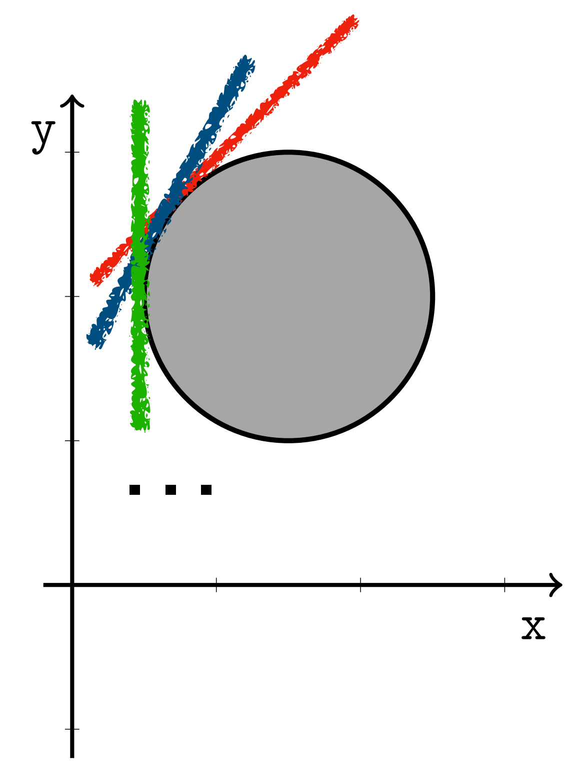
(b) Concretization of a_1

(c) Concretization of a_2

Figure: Convex polyhedra abstraction

Best Abstraction is Not Always Obtainable

- Computing the best abstraction is expensive in general, or sometimes even impossible.
  - In case of the diameter, there is no best abstraction since it requires infinitely many linear inequalities.



(a) A concrete set

- Thus in practice, we often use abstractions as precise as possible but may not be the best.

Reachable States of the Example Program

Example

```
init([0,1] × [0,1]);
iter{
    {
        translation(1,0)
    }or{
        translation(0.5,0.5)
    }
}
```

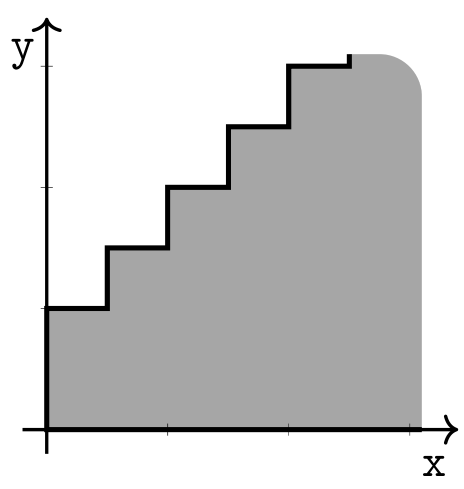
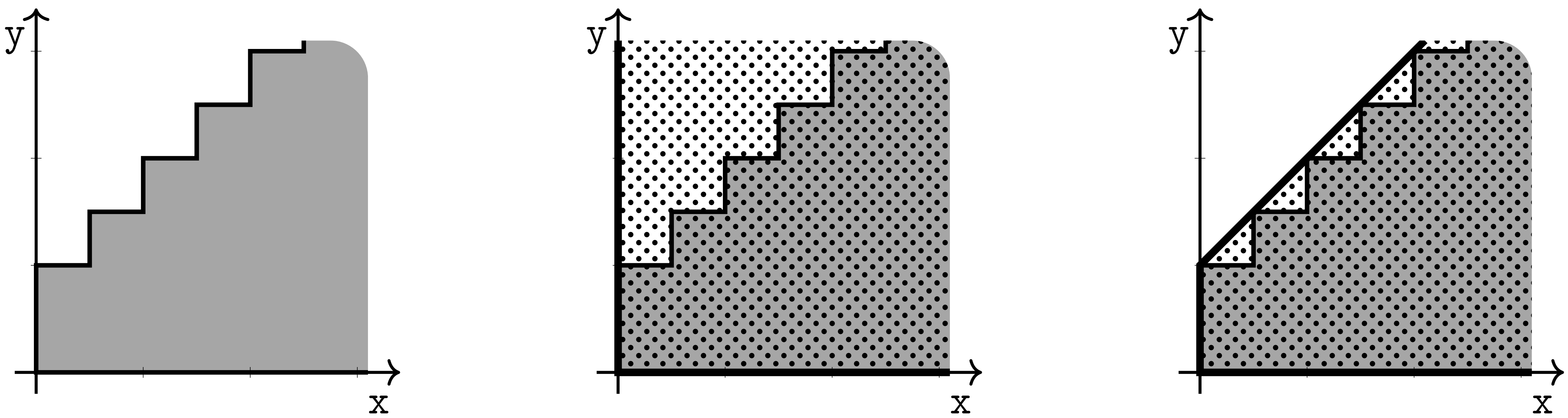


Figure: Reachable states

Abstractions of the Semantics of the Example Program



(a) Reachable states (b) Intervals abstraction (c) Convex polyhedra abstraction

Figure: Program's reachable states and abstraction

Abstract Semantics Computation

Recall the example language

$\mathtt{p}$	$::=$	$\mathtt{init}(\mathfrak{R})$	initialization, with a state in $\mathfrak{R}$
	$\mid$	$\mathtt{translation}(u, v)$	translation by vector (u, v)
	$\mid$	$\mathtt{rotation}(u, v, \theta)$	rotation defined by center (u, v) and angle θ
	$\mid$	$\mathtt{p}\ ;\ \mathtt{p}$	sequence of operations
	$\mid$	$\{\mathtt{p}\}\mathtt{or}\{\mathtt{p}\}$	non-deterministic choice
	$\mid$	$\mathtt{iter}\{\mathtt{p}\}$	non-deterministic iterations

Approach

A sound analysis for a program is constructed by computing sound abstract semantics of the program's components.

Sound Analysis Function for the Example Language

- Input: a program $\mathtt{p}$ and an abstract area a (pre-state)
- Output: an abstract area a' (post-state)

Definition (sound analysis)

An `analysis` is sound if and only if **it captures the real executions of the input program**.

If an execution of $\mathtt{p}$ moves a point $(\mathtt{x}, \mathtt{y})$ to point $(\mathtt{x}', \mathtt{y}')$,
then for all abstract element a such that $(\mathtt{x}, \mathtt{y}) \in \gamma(a)$,

$$(\mathtt{x}', \mathtt{y}') \in \gamma(\mathtt{analysis}(\mathtt{p}, a))$$

Sound Analysis Function as a Diagram

If

$$\begin{tikzcd} a_{\text{pre}} & \\ (\mathtt{x}, \mathtt{y}) \arrow[u, "\text{abstraction}"] \arrow[r, "\text{run } \mathtt{p}"] & (\mathtt{x}', \mathtt{y}') \end{tikzcd}$$

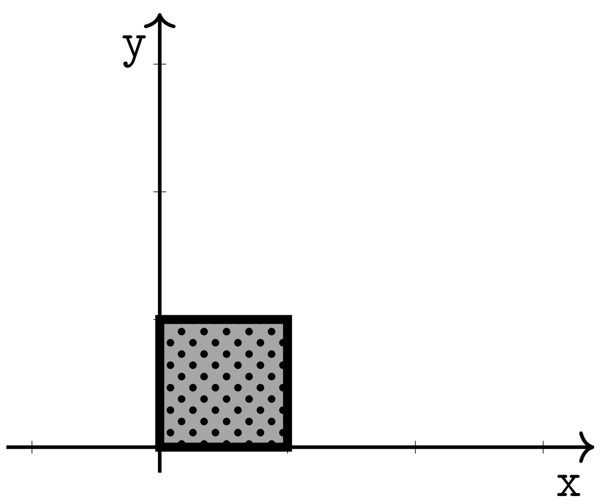
then

$$\begin{tikzcd} a_{\text{pre}} \arrow[r, "\text{analyze } \mathtt{p}"] & a_{\text{post}} = \mathtt{analysis}(\mathtt{p}, a_{\text{pre}}) \\ (\mathtt{x}, \mathtt{y}) \arrow[u, "\text{abstraction}"] \arrow[r, "\text{run } \mathtt{p}"] & (\mathtt{x}', \mathtt{y}') \arrow[u, "\text{abstraction}"'] \end{tikzcd}$$

Figure: Sound analysis of a program $\mathtt{p}$

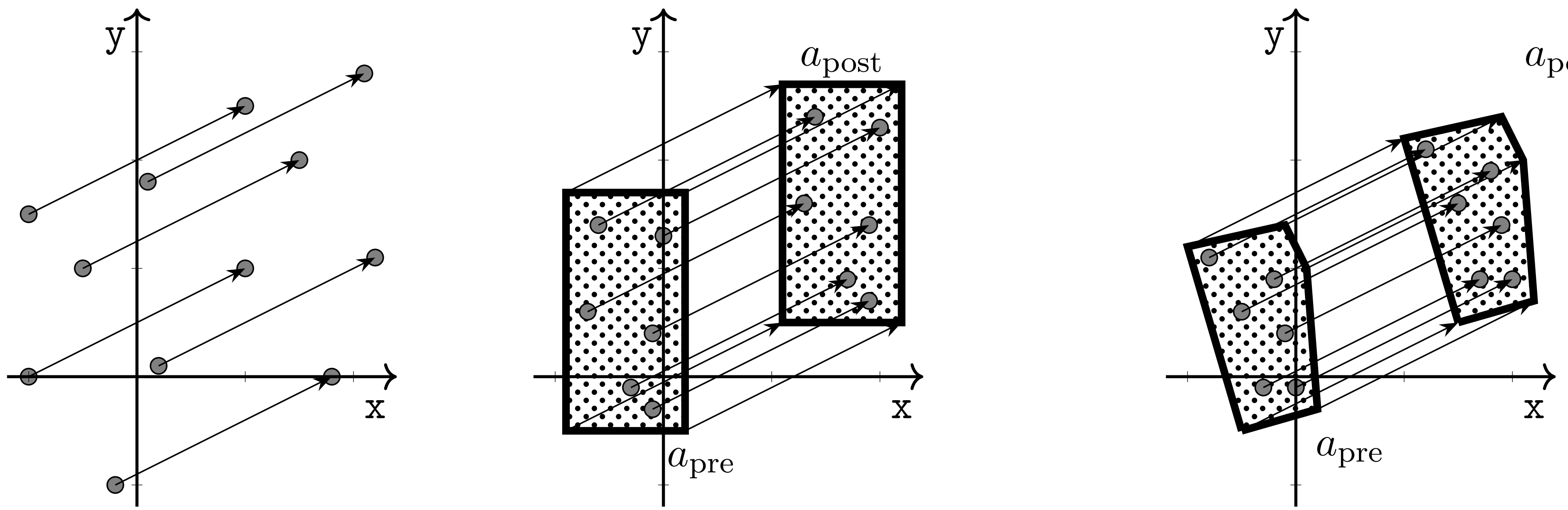
Abstract Semantics Computation: init(R)

- Select, if any, the best abstraction of the region $\mathfrak{R}$.
- For the example program with the intervals or convex polyhedra abstract domains, analysis of $\mathtt{init}([0,1] \times [0,1])$ is



$$\mathtt{analysis}(\mathtt{init}(\mathfrak{R}), a) = \text{best abstraction of the region } \mathfrak{R}$$

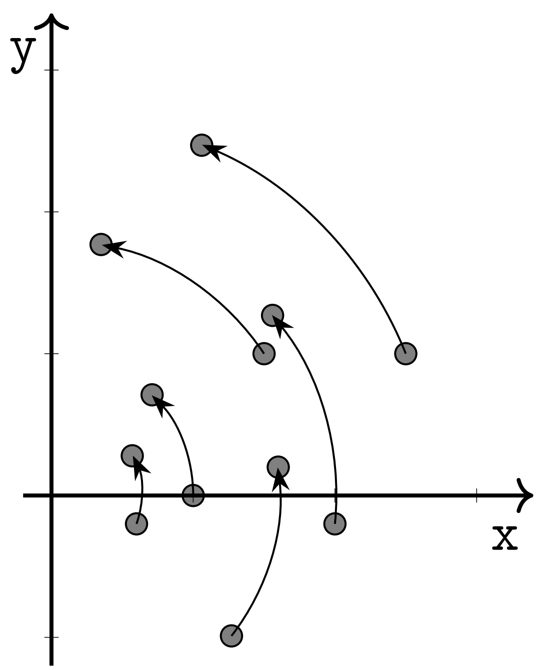
Abstract Semantics Computation: translation(u, v)



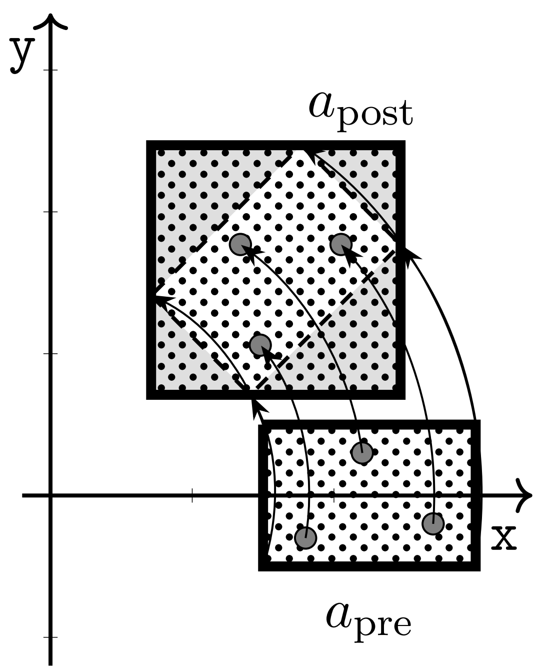
(a) Concrete semantics　　(b) Intervals　　(c) Convex polyhedra

$$\texttt{analysis}(\texttt{translation}(u, v), a) = \begin{cases} \text{return an abstract state that contains} \\ \text{the translation of } a \end{cases}$$

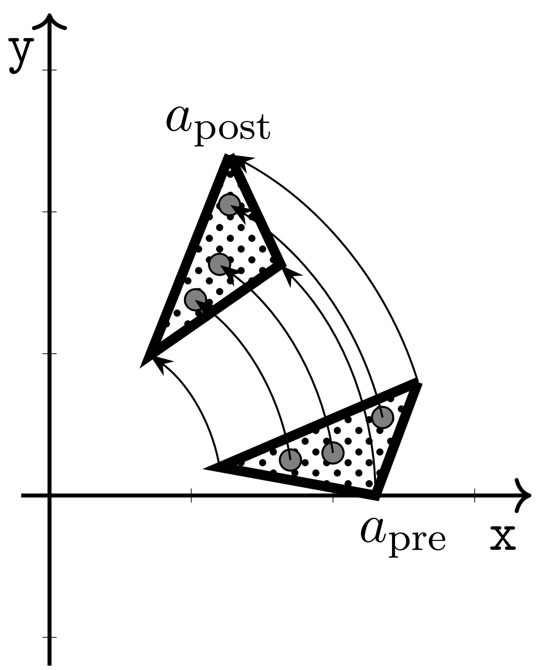
Abstract Semantics Computation: rotation(u, v, θ)



(d) Concrete semantics



(e) Intervals



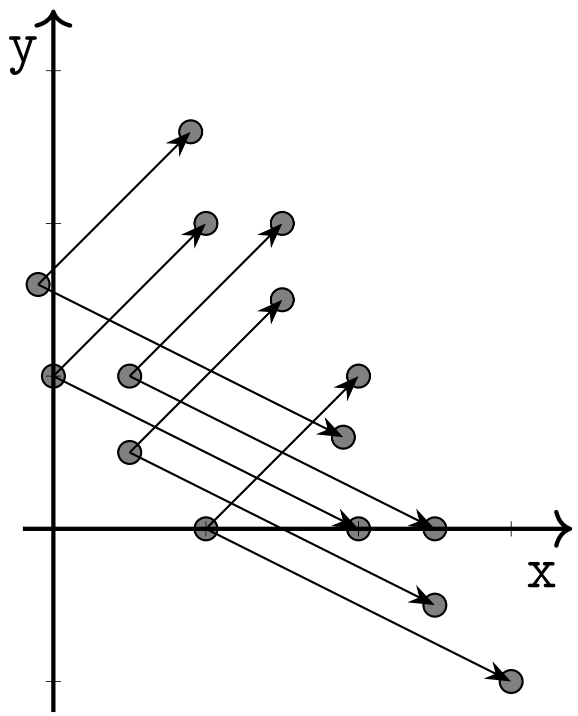
(f) Convex polyhedra

$$\mathtt{analysis}(\mathtt{rotation}(u, v, \theta), a) = \begin{cases} \text{return an abstract state that contains} \\ \text{the rotation of } a \end{cases}$$

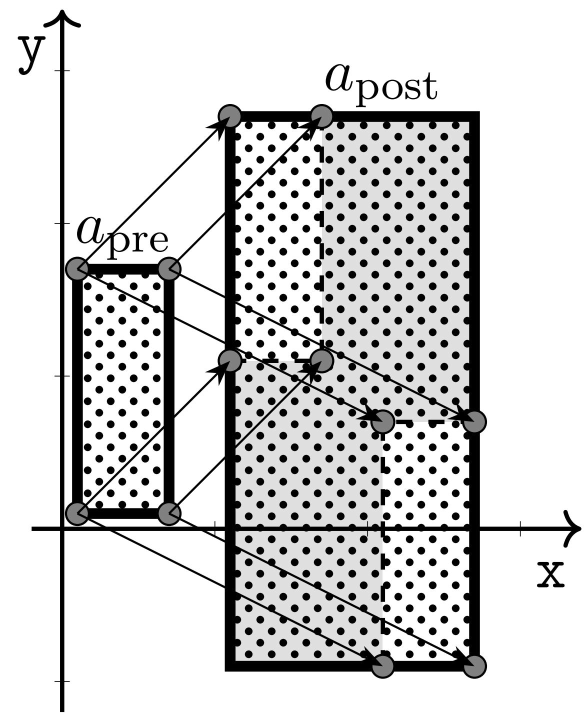
Abstract Semantics Computation: $p_0; p_1$

$$\texttt{analysis}(\mathtt{p}_0;\mathtt{p}_1, a) = \texttt{analysis}(\mathtt{p}_1, \texttt{analysis}(\mathtt{p}_0, a))$$

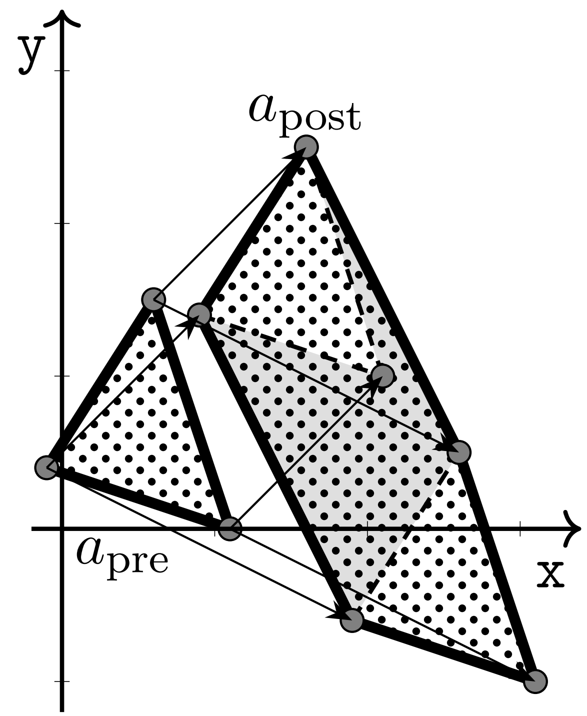
Abstract Semantics Computation: {p}or{p}



(g) Concrete semantics



(h) Intervals



(i) Convex polyhedra

$$\mathtt{analysis}(\{\mathtt{p}_0\}\mathtt{or}\{\mathtt{p}_1\}, a) = \mathtt{union}(\mathtt{analysis}(\mathtt{p}_1, a), \mathtt{analysis}(\mathtt{p}_0, a))$$

Abstract Semantics Computation: iter{b}

$$
p ::= \left\{ \begin{array}{l} \texttt{\textbf{iter}}\{ \\ \qquad \texttt{b} \\ \} \end{array} \right. \quad \equiv \quad \begin{array}{l} \{\} \\ \texttt{\textbf{or}}\{\texttt{b}\} \\ \texttt{\textbf{or}}\{\texttt{b};\texttt{b}\} \\ \texttt{\textbf{or}}\{\texttt{b};\texttt{b};\texttt{b}\} \\ \texttt{\textbf{or}}\{\texttt{b};\texttt{b};\texttt{b};\texttt{b}\} \\ \vdots \end{array}
$$

Abstract Semantics Computation: iter{p}

$$
\begin{array}{llll}
\text{program} & \mathrm{p}_0 & \text{is} & \{\} \\
\text{program} & \mathrm{p}_1 & \text{is} & \{\}\,\mathtt{\textbf{or}}\,\{\mathtt{b}\} \\
\text{program} & \mathrm{p}_2 & \text{is} & \{\}\,\mathtt{\textbf{or}}\,\{\mathtt{b}\}\,\mathtt{\textbf{or}}\,\{\mathtt{b};\mathtt{b}\} \\
\text{program} & \mathrm{p}_3 & \text{is} & \{\}\,\mathtt{\textbf{or}}\,\{\mathtt{b}\}\,\mathtt{\textbf{or}}\,\{\mathtt{b};\mathtt{b}\}\,\mathtt{\textbf{or}}\,\{\mathtt{b};\mathtt{b};\mathtt{b}\} \\
 & & \vdots &
\end{array}
$$

$$\boxed{\mathrm{p}_{k+1} \quad \text{is equivalent to} \quad \mathrm{p}_k\,\mathtt{\textbf{or}}\,\{\mathrm{p}_k;\mathtt{b}\}}$$

Therefore,

$$\mathtt{analysis}(\mathrm{p}_{k+1}, a) = \mathtt{union}(\mathtt{analysis}(\mathrm{p}_k, a), \mathtt{analysis}(\mathtt{b}, \mathtt{analysis}(\mathrm{p}_k, a)))$$

Abstract Semantics Computation: iter{p}

$$\texttt{analysis}(\textbf{\texttt{iter}}\{\texttt{p}\}, a) \quad = \quad \begin{cases} \texttt{R} \leftarrow a; \\ \text{repeat} \\ \qquad \texttt{T} \leftarrow \texttt{R}; \\ \qquad \texttt{R} \leftarrow \texttt{union}(\texttt{R}, \texttt{analysis}(\texttt{p}, \texttt{R})) \\ \text{until } \texttt{inclusion}(\texttt{R}, \texttt{T}) \\ \text{return } \texttt{T}; \end{cases}$$

operator `inclusion` returns **true** only when it succeeds checking inclusion

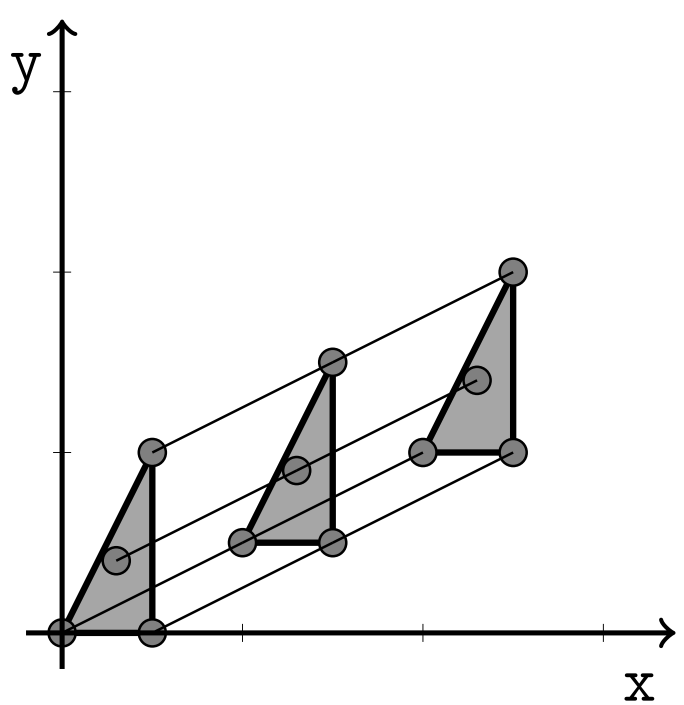
Abstract Semantics Computation: iter{p} Example: Sign Abstraciton

$$\texttt{init}(\{(\mathtt{x},\mathtt{y}) \mid 0 \le \mathtt{y} \le 2\mathtt{x} \textit{ and } \mathtt{x} \le 0.5\});$$

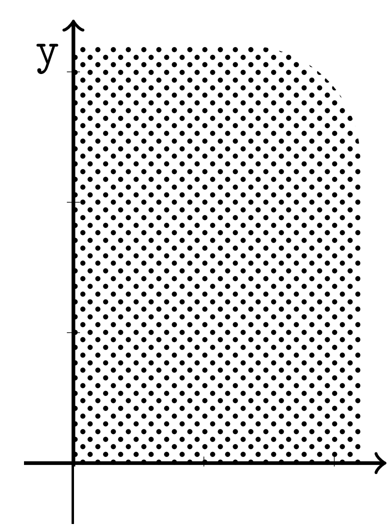
$$\texttt{iter}\{$$

$$\quad \texttt{translation}(1, 0.5)$$

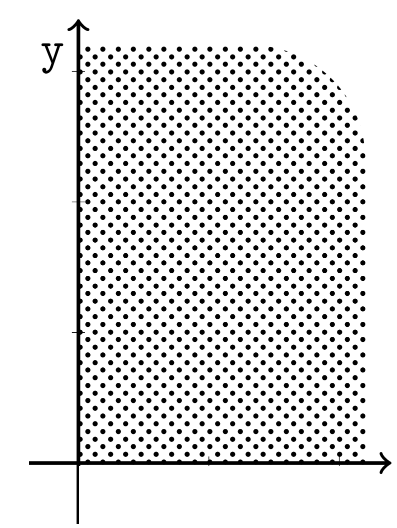
$$\}$$



(a) Concrete semantics



(b) Analysis of $\mathtt{p}_0$ (0 iteration)

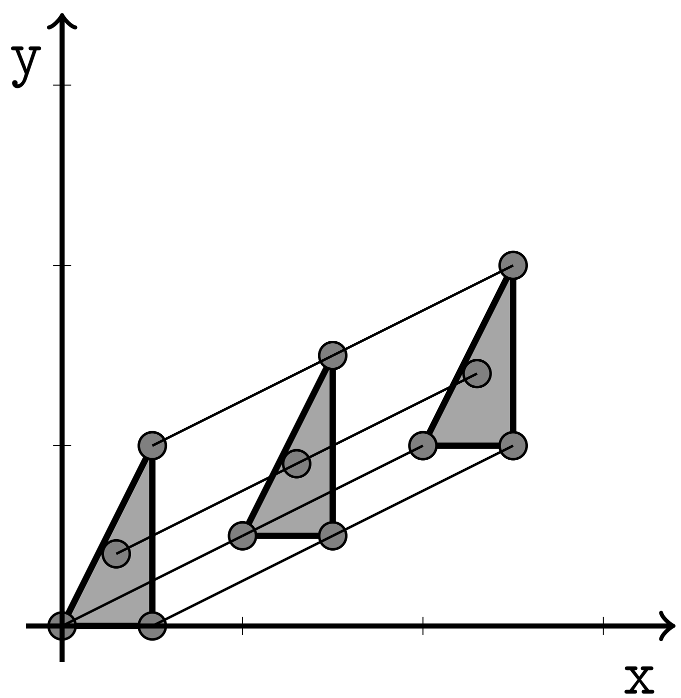


(c) Analysis of $\mathtt{p}_1$ (up to 1 iteration)

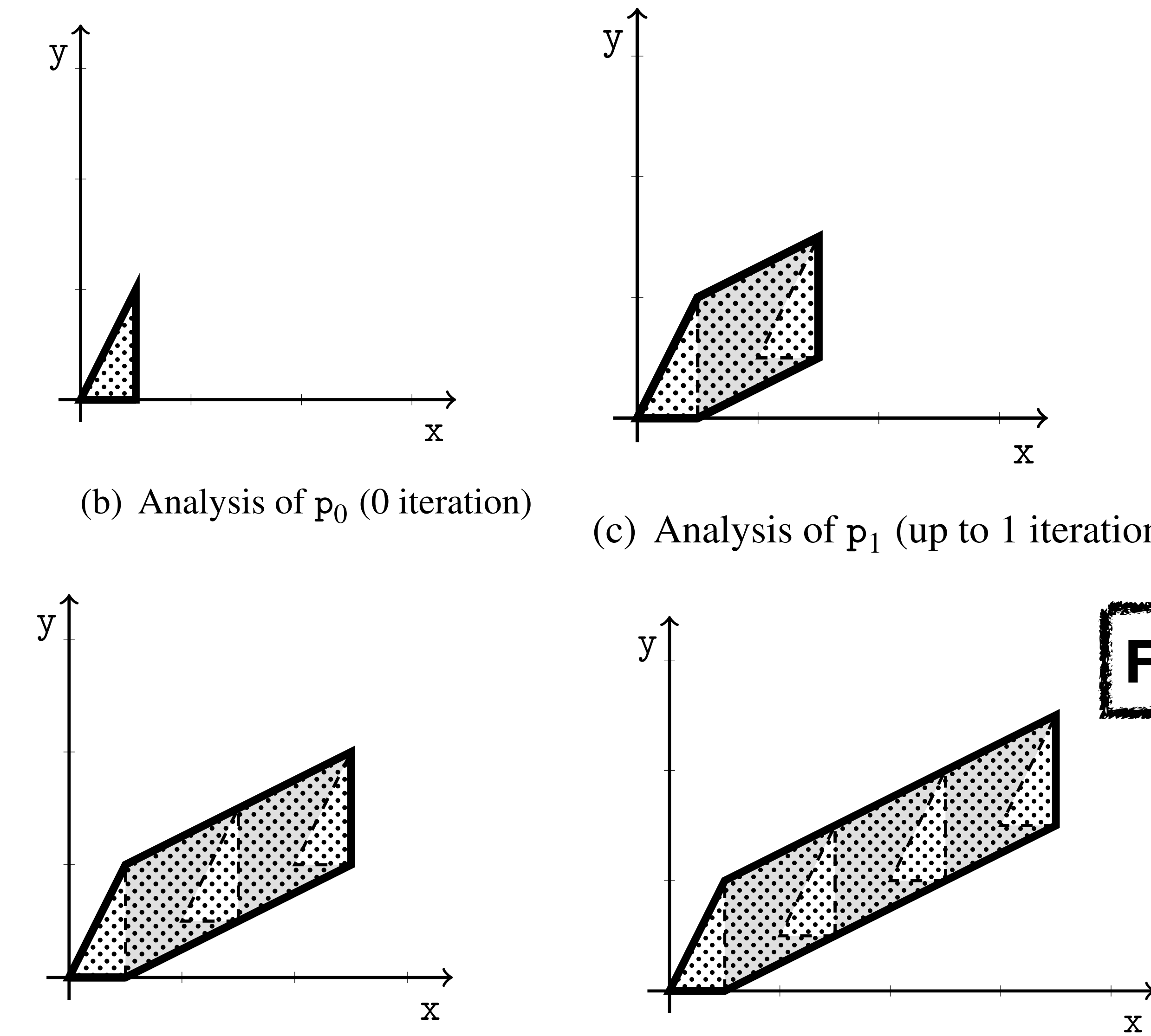
No change. Done!

Abstract Semantics Computation: iter{p} Example: Convex Polyhedra

$$\texttt{init}(\{(x,y) \mid 0 \le y \le 2x \textit{ and } x \le 0.5\});$$
$$\texttt{iter}\{$$
$$\quad \texttt{translation}(1, 0.5)$$
$$\}$$



(a) Concrete semantics



(b) Analysis of p_0 (0 iteration)

(c) Analysis of p_1 (up to 1 iteration)

(d) Analysis of p_2 (up to 2 iterations)

(e) Analysis of p_3 (up to 3 iterations)

Forever!

...

Abstract Semantics Computation: iter{p} Example: Convex Polyhedra

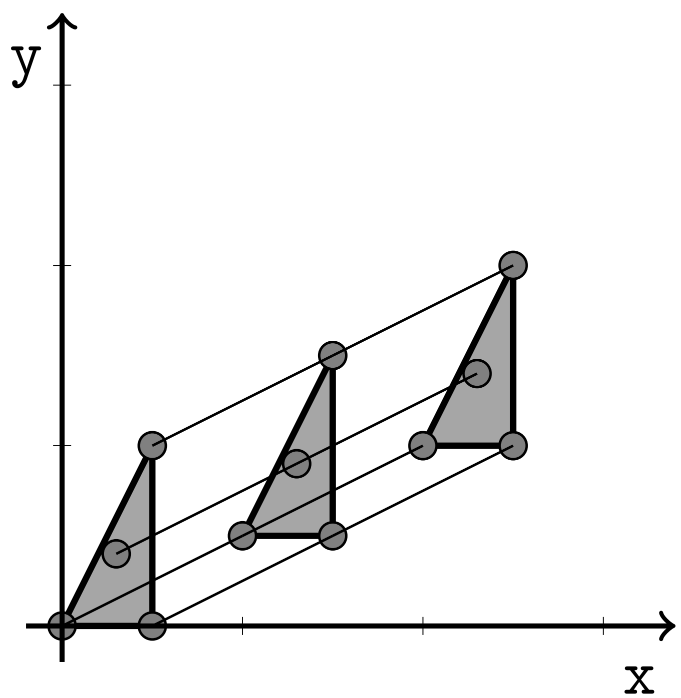
$\texttt{init}(\{(\mathtt{x},\mathtt{y}) \mid 0 \leq \mathtt{y} \leq 2\mathtt{x} \textit{ and } \mathtt{x} \leq 0.5\});$

$\texttt{iter}\{$

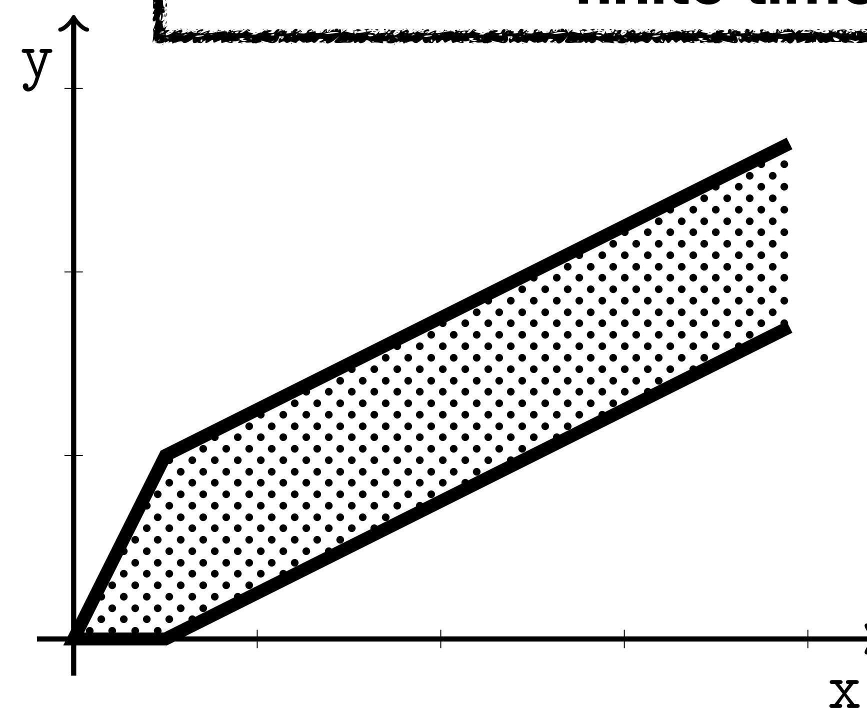
$\quad \texttt{translation}(1, 0.5)$

$\}$

We want to converge to this state in finite time!



(a) Concrete semantics



(f) Expected result

Widening

- To ensure termination of the analysis, we need to *enforce* the convergence of the iterations.
- In case of convex polyhedra
    - An abstract element = (finitely many) inequalities
    - If we decrease the number of inequalities at each iteration, it will eventually terminate.

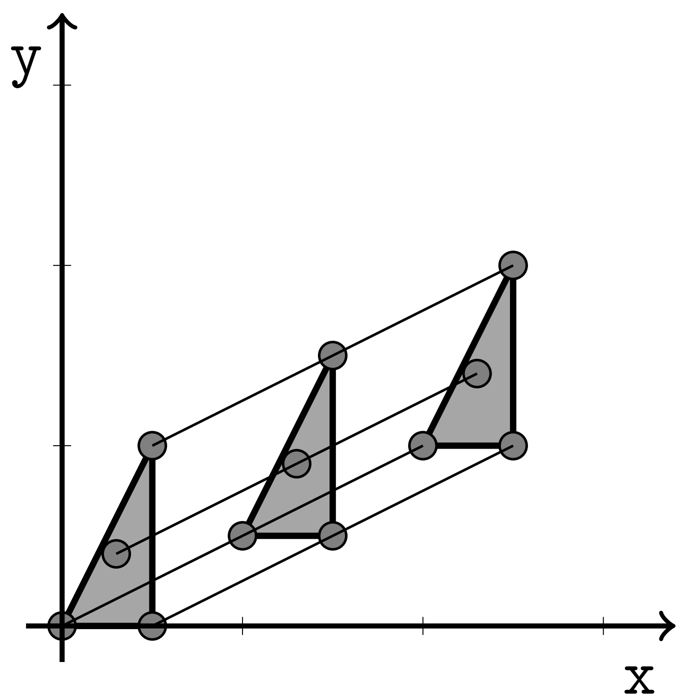
Abstract Semantics Computation: iter{p}
Example: Convex Polyhedra

$\mathtt{init}(\{(\mathtt{x},\mathtt{y}) \mid 0 \leq \mathtt{y} \leq 2\mathtt{x} \textit{ and } \mathtt{x} \leq 0.5\});$

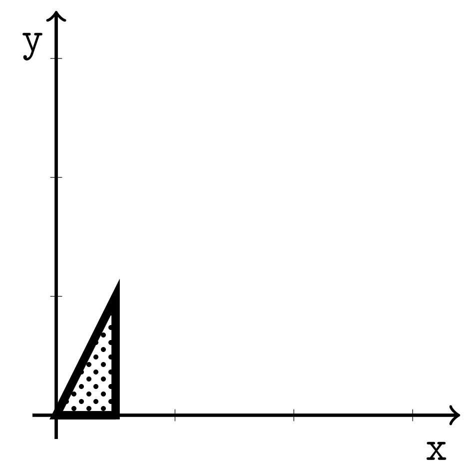
$\mathtt{iter}\{$

$\quad \mathtt{translation}(1, 0.5)$

$\}$

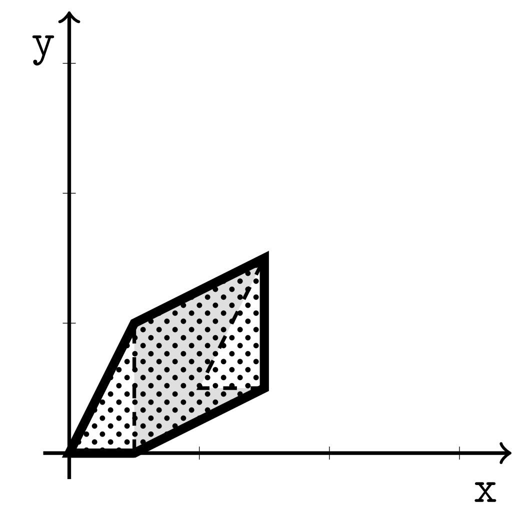


(a) Concrete semantics



(b) Analysis of $\mathtt{p}_0$ (0 iteration)

$$\boxed{\begin{array}{l} 0 \leq y \\ y \leq 2x \\ x \leq 0.5 \end{array}}$$

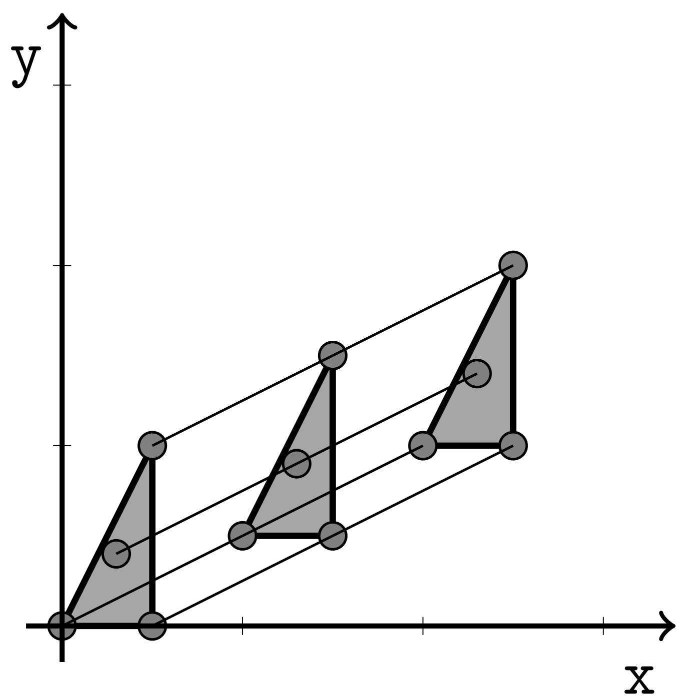


(c) Analysis of $\mathtt{p}_1$ (up to 1 iteration)

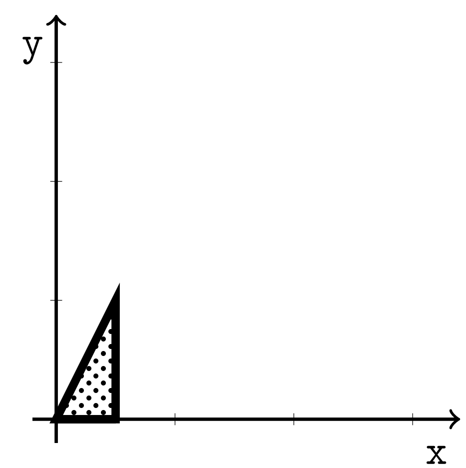
$$\boxed{\begin{array}{l} 0 \leq y \\ y \leq 2x \\ x \leq 1.5 \\ y \leq 0.5x + 1 \\ y \geq 0.5x - 1 \end{array}}$$

Abstract Semantics Computation: iter{p} Example: Convex Polyhedra

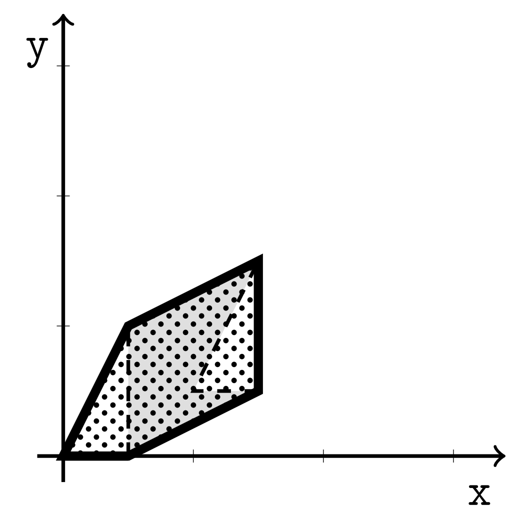
```
init({(x,y) | 0 ≤ y ≤ 2x and x ≤ 0.5});
iter{
    translation(1,0.5)
}
```



(a) Concrete semantics



(b) Analysis of $\mathtt{p}_0$ (0 iteration)



(c) Analysis of $\mathtt{p}_1$ (up to 1 iteration)

Remained

$0 \leq y$

$y \leq 2x$

Modified

$x \leq 0.5$

$0 \leq y$

$y \leq 2x$

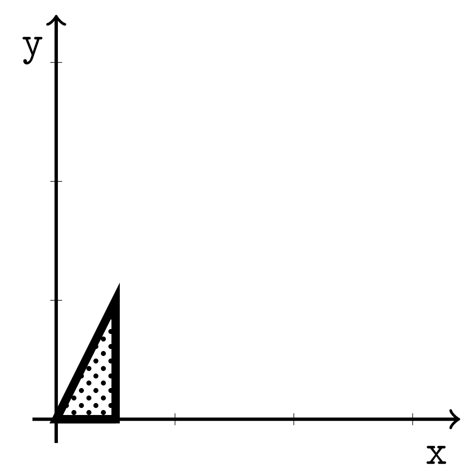
$x \leq 1.5$

$y \leq 0.5x + 1$

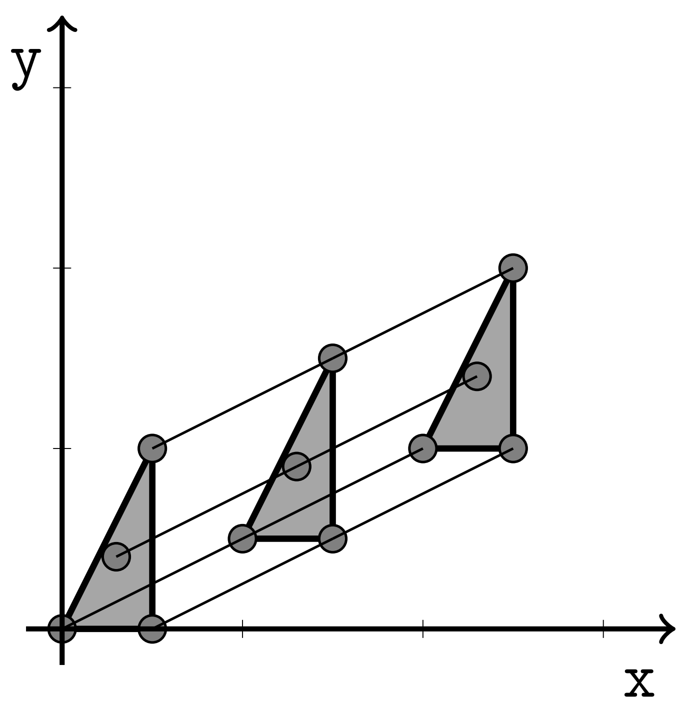
$y \geq 0.5x - 1$

Abstract Semantics Computation: iter{p} Example: Convex Polyhedra

$$
\begin{aligned}
&\texttt{init}(\{(\mathtt{x},\mathtt{y}) \mid 0 \leq \mathtt{y} \leq 2\mathtt{x} \textit{ and } \mathtt{x} \leq 0.5\});\\
&\texttt{iter}\{\\
&\qquad \texttt{translation}(1, 0.5)\\
&\}
\end{aligned}
$$



(b) Analysis of p_0 (0 iteration)



(a) Concrete semantics

$0 \leq y$

$y \leq 2x$

$\cancel{x \leq 0.5}$

Discard this

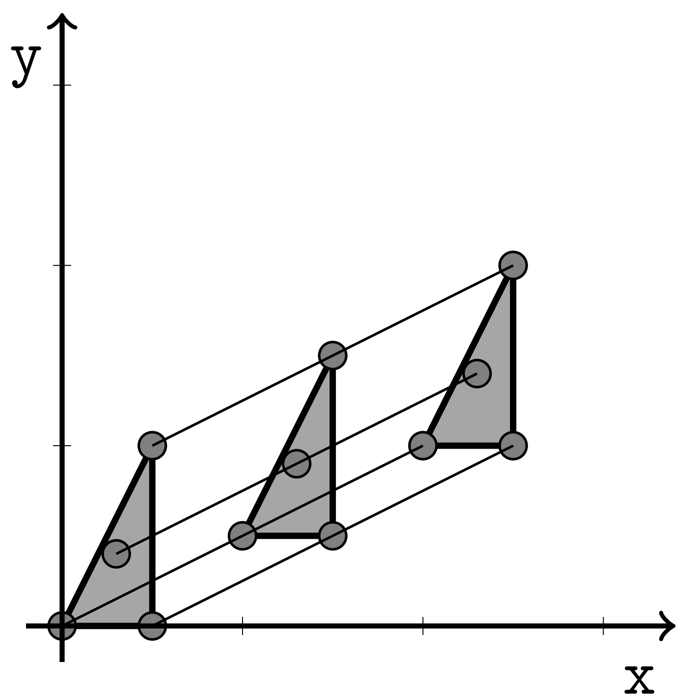
Abstract Semantics Computation: iter{p}
Example: Convex Polyhedra

$\texttt{init}(\{(\mathtt{x},\mathtt{y}) \mid 0 \leq \mathtt{y} \leq 2\mathtt{x} \textit{ and } \mathtt{x} \leq 0.5\});$

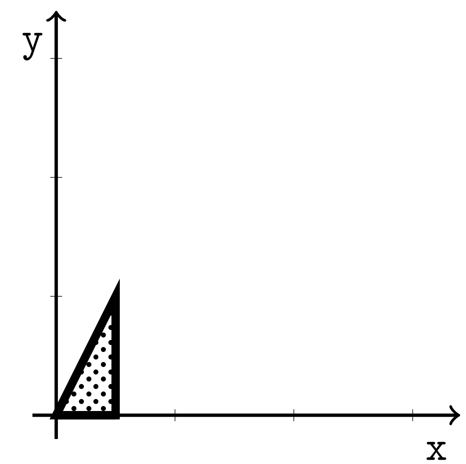
$\texttt{iter}\{$

$\quad \texttt{translation}(1, 0.5)$

$\}$



(a) Concrete semantics

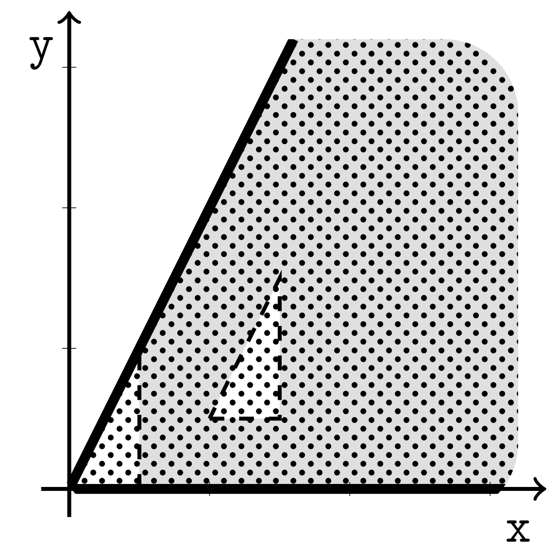


(b) Analysis of p_0 (0 iteration)

$0 \leq y$

$y \leq 2x$

$x \leq 0.5$



(b) Iteration 1

$0 \leq y$

$y \leq 2x$

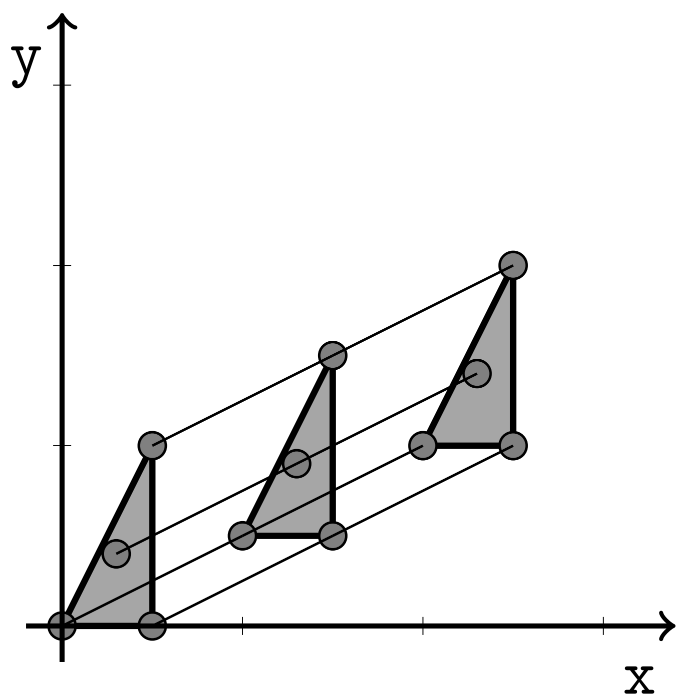
Abstract Semantics Computation: iter{p}
Example: Convex Polyhedra

$\texttt{init}(\{(\mathrm{x},\mathrm{y}) \mid 0 \leq \mathrm{y} \leq 2\mathrm{x} \textit{ and } \mathrm{x} \leq 0.5\});$

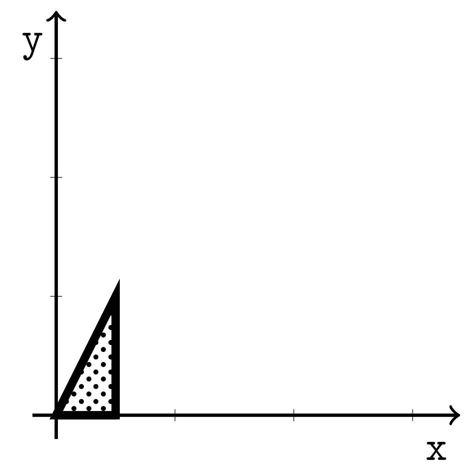
$\texttt{iter}\{$

$\quad \texttt{translation}(1, 0.5)$

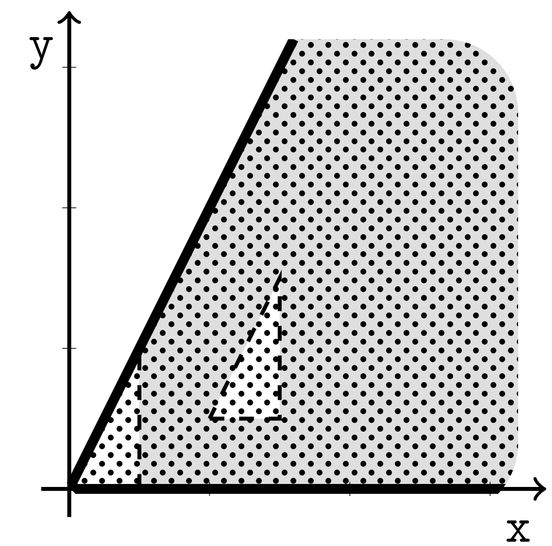
$\}$



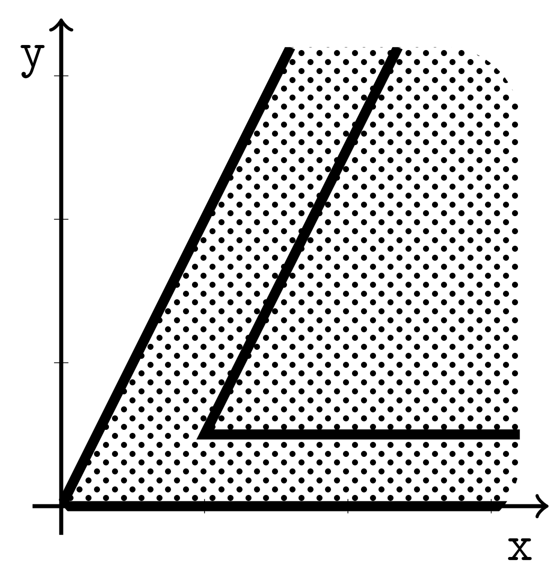
(a) Concrete semantics



(b) Analysis of p_0 (0 iteration)



(b) Iteration 1



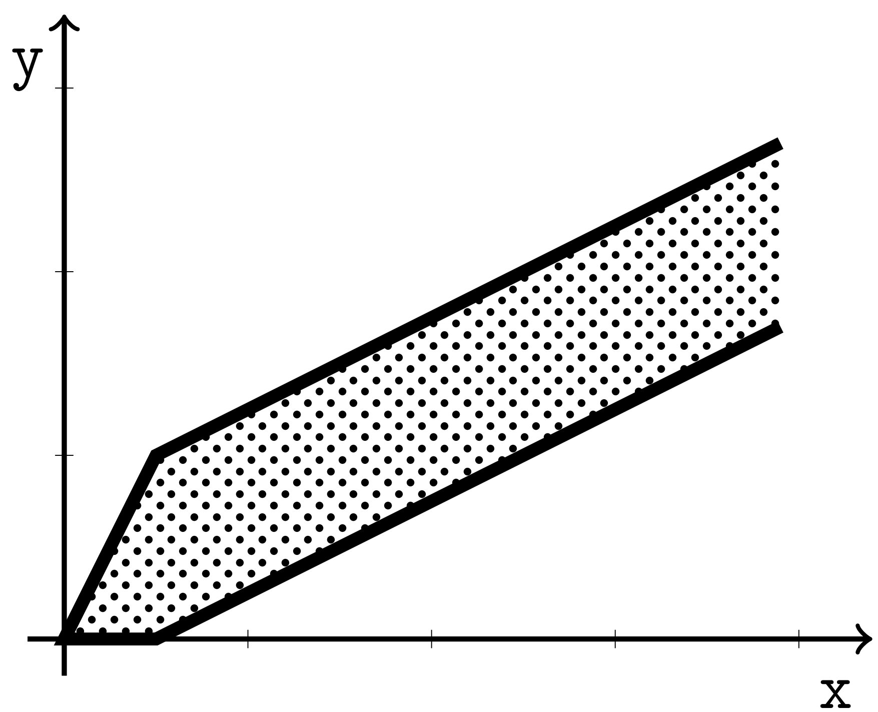
(c) Iteration 2 and limit

Imprecision due to Widening

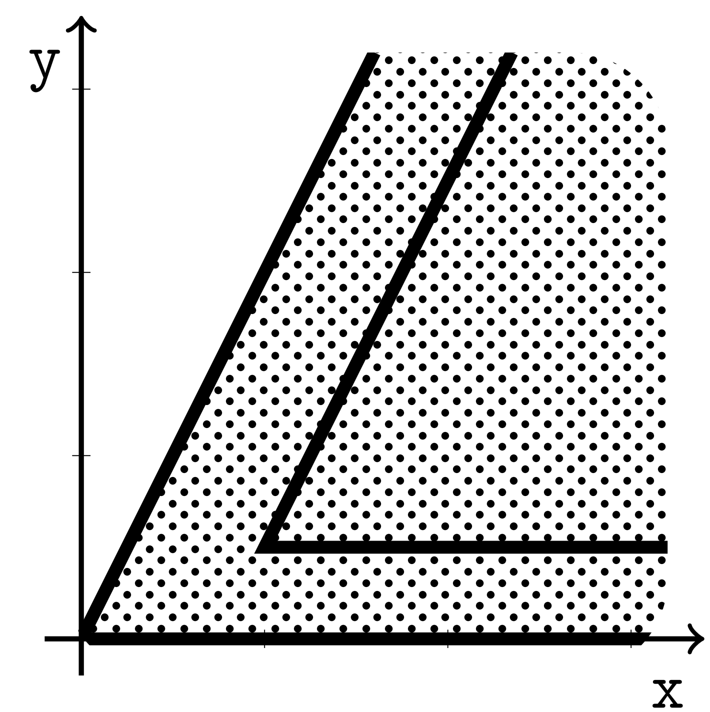
$$\text{operator } \texttt{widen} \begin{cases} \text{over-approximates unions} \\ \text{enforces convergence} \end{cases}$$

- Widening guarantees termination of the analysis.

- However, it incurs significant precision loss.



(f) Expected result



(c) Iteration 2 and limit

Abstract Iteration with Widening

Recall

$$\begin{aligned} \mathtt{iter}\{\mathtt{p}\} &= \{\}\ \mathtt{or}\ \{\mathtt{p}\}\ \mathtt{or}\ \{\mathtt{p};\mathtt{p}\}\ \mathtt{or}\ \cdots \\ &= \lim_i \mathtt{p}_i \end{aligned}$$

where

$$\mathtt{p}_0 = \{\} \qquad \mathtt{p}_{k+1} = \mathtt{p}_k\ \mathtt{or}\ \{\mathtt{p}_k;\mathtt{p}\}$$

Hence,

$$\mathtt{analysis}(\mathtt{iter}\{\mathtt{p}\}, a) = \begin{cases} \mathtt{R} \leftarrow a; \\ \text{repeat} \\ \qquad \mathtt{T} \leftarrow \mathtt{R}; \\ \qquad \mathtt{R} \leftarrow \mathtt{widen}(\mathtt{R}, \mathtt{analysis}(\mathtt{p}, \mathtt{R})); \\ \text{until } \mathtt{inclusion}(\mathtt{R}, \mathtt{T}) \\ \text{return } \mathtt{T}; \end{cases}$$

$$\text{operator } \mathtt{widen} \quad \begin{cases} \text{over approximates unions} \\ \text{enforces finite convergence} \end{cases}$$

Loop Unrolling for Precision Improvement

$$\texttt{init}(\{(\mathtt{x},\mathtt{y}) \mid 0 \leq \mathtt{y} \leq 2\mathtt{x}\ \mathit{and}\ \mathtt{x} \leq 0.5\});$$

$\texttt{iter}\{$

    $\texttt{translation}(1, 0.5)$

$\}$

Loop unrolling once

$$\texttt{init}(\{(\mathtt{x},\mathtt{y}) \mid 0 \leq \mathtt{y} \leq 2\mathtt{x}\ \mathit{and}\ \mathtt{x} \leq 0.5\});$$

$\{\}\texttt{or}\{$

    $\texttt{translation}(1, 0.5)$

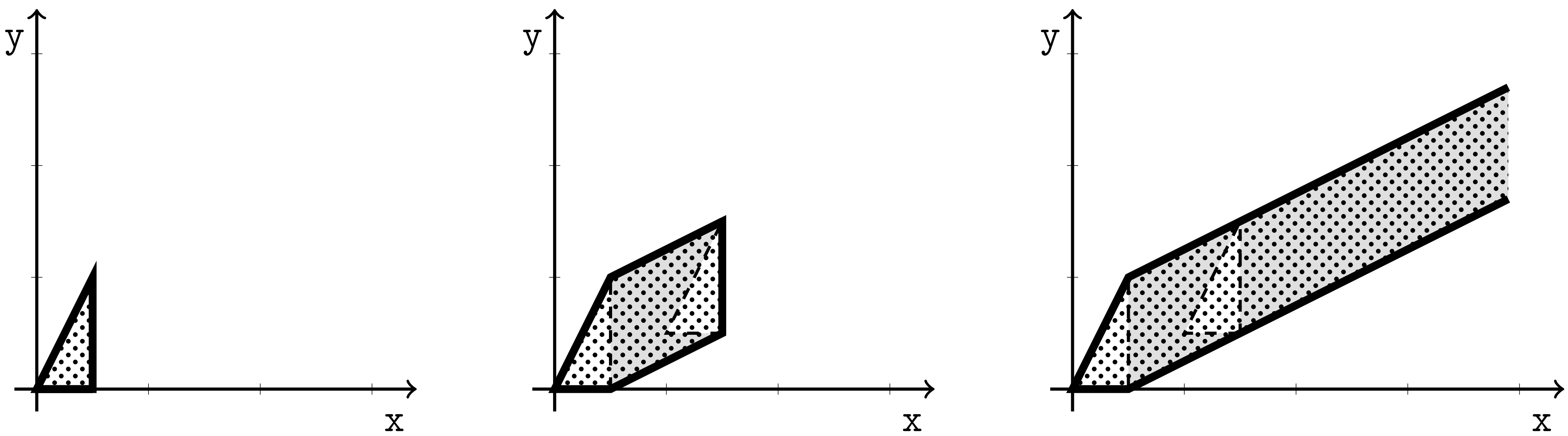
$\}$

$\texttt{iter}\{$

    $\texttt{translation}(1, 0.5)$

$\}$

Loop Unrolling for Precision Improvement



(a) Iteration 0 (b) Iteration 1, `union` (c) Iteration 2, `widen`, limit

Figure 2.17

Abstract iteration with widening and unrolling

Abstract Semantics Function analysis At a Glance

The $\texttt{analysis}(\texttt{p}, a)$ is finitely computable and sound.

$$
\begin{array}{rcl}
\texttt{analysis}(\texttt{init}(\mathfrak{R}), a) & = & \text{best abstraction of the region } \mathfrak{R} \\
\texttt{analysis}(\texttt{translation}(u, v), a) & = & \left\{ \begin{array}{l} \text{return an abstract state that contains} \\ \text{the translation of } a \end{array} \right. \\
\texttt{analysis}(\texttt{rotation}(u, v, \theta), a) & = & \left\{ \begin{array}{l} \text{return an abstract state that contains} \\ \text{the rotation of } a \end{array} \right. \\
\texttt{analysis}(\{\texttt{p}_0\}\texttt{or}\{\texttt{p}_1\}, a) & = & \texttt{union}(\texttt{analysis}(\texttt{p}_1, a), \texttt{analysis}(\texttt{p}_0, a)) \\
\texttt{analysis}(\texttt{p}_0; \texttt{p}_1, a) & = & \texttt{analysis}(\texttt{p}_1, \texttt{analysis}(\texttt{p}_0, a)) \\
\texttt{analysis}(\texttt{iter}\{\texttt{p}\}, a) & = & \left\{ \begin{array}{l} \texttt{R} \leftarrow a; \\ \text{repeat} \\ \qquad \texttt{T} \leftarrow \texttt{R}; \\ \qquad \texttt{R} \leftarrow \texttt{widen}(\texttt{R}, \texttt{analysis}(\texttt{p}, \texttt{R})); \\ \text{until } \texttt{inclusion}(\texttt{R}, \texttt{T}) \\ \text{return } \texttt{T}; \end{array} \right.
\end{array}
$$

Soundness of Abstract Semantics Function analysis

Sound analysis

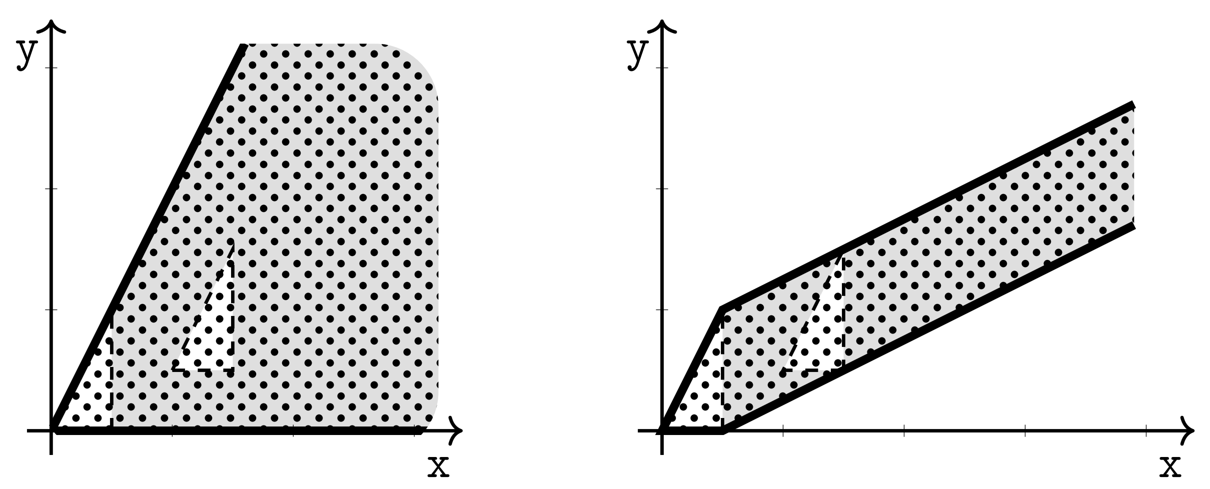
If an execution of p from a state (x, y) generates the state $(\mathrm{x}', \mathrm{y}')$, then for all abstract element a such that $(\mathrm{x}, \mathrm{y}) \in \gamma(a)$,

$$(\mathrm{x}', \mathrm{y}') \in \gamma(\texttt{analysis}(\mathrm{p}, a))$$

Theorem. The **analysis** function is sound.

Verification of the Property of Interest

- Does program compute a point inside no-fly zone $\mathfrak{D}$?
- Need to collect the set of reachable points.
- Run analysis$(\mathrm{p}, -)$ and collect all points $\mathfrak{R}$ from every call to analysis.
- Since analysis is sound, the result is an over approx. of the reachable points.
- **If $\mathfrak{R} \cap \mathfrak{D} = \emptyset$, then p is verified. Otherwise, we don't know.**



(a) An example $\mathfrak{R}$ (b) A more precise $\mathfrak{R}$