Solvers Principles and Architecture (SPA)

Part 2

SMT Solvers

Master Sciences Informatique (Sif)
September, 2018
Rennes

Khalil Ghorbal
khalil.ghorbal@inria.fr

Outline

1 First Order Logic

2 Static Analysis-based Abstract Interpretation

Recall that **logic** is a pair of **syntax** and **semantics**.

Syntax

- Alphabet: set of symbols
- Expressions: sequences of symbols
- Rules: identifying **well-formed** expressions

Semantics

- **Meaning**: what is meant by well-formed expressions
- Rules: infer the meaning from subexpressions

In addition to Logical symbols: $\neg$, $\wedge$, $\longrightarrow$, etc. (alphabet of propositional logic)
We will be adding:

- variables symbols: x, y, etc.
- **parameters**, or non-logical symbols: $\exists$, f, $\leq$, $=$, $+$, π, etc.

Quantifiers

- Exists: $\exists$
- Forall: $\forall$

Functions

- Symbol (or name)
- Output type (or kind) – (Co-domain)
- Inputs arity (or cardinality) and their respective types – (Domain)

Predicates

- Sets described by some relations
- n-arity functions with co-domain $\{\mathtt{F}, \mathtt{T}\}$ (False/True in PL)
- Predicate symbols: $=$, $<$, $\in$, etc.

Constants

- Functions with **arity zero**
- Usual symbols: π, 1, $\emptyset$, etc.
- Predicates with arity zero are the propositional constants ($\mathtt{F}$, $\mathtt{T}$).

First-order means **quantifiers** are only allowed over **variables**: $Q_i x_i$.

- **Each quantifier is necessarily related to a variable**.
- A variable is either **free** or bound by a quantifier.

Examples

- Function $+ : (x, y) \mapsto x + y$
- Predicate: $f(x) = f(y)$ (for some function f)
- Predicate: $x \leq f(y)$

Example of First-order Languages

Basic Set Language

- Relationship predicate: $\mathcal{R}$
- Constant: $\emptyset$

Elementary Number Language

- Constant: 0
- Function: $\mathtt{Succ}$
- Equality predicate: $=$

Terms

Built **inductively** from functions' symbols **applied** to constants and variables.

- A variable v is a term
- A constant 0 is a term
- The function f applied to terms t_1 and t_2 is a term named $f(t_1, t_2)$

Atomic Formulas

Built by applying **predicates on terms**.

- $\mathtt{F}/\mathtt{T}$ are atomic predicates
- $\leq\ v\,0$ is an atomic predicate (prefix notation)
- $t_1 = t_2$ is an atomic predicate (infix notation)

Built **inductively** from atomic formulas with logic connectives and quantifiers.

- $\neg\phi$ is a formula
- $\phi_1 \longrightarrow \phi_2$ is a formula
- $Q_1 v_1 . Q_2 v_2 . \phi(t, g(t))$ is a formula
- Terms t and $g(t)$ may or may not contain the variables v_1 and v_2

Free and Bound Variables

A variable in a wff is either **free** or bound to a quantifier.

- $\exists v_1.f(v_1) < v_2$: v_2 is free
- $\forall v_1.\exists v_2.P(v_1, g(v_1, v_2))$: both variables are bound

A wff with no free variables is called a **sentence**.

A **signature** (Σ) contains the parameters of the language, that is all its non-logical **symbols**: constants, functions, and predicates.

Example: Elementary Numbers Signatures

- $(0, \mathtt{Succ}, =)$
- $(0, 1, +, -, >)$

An **interpretation** (M) of a signature is twofold:

- An underlying domain $\mathcal{D}_M$ (e.g. natural numbers)
- An interpretation of all the symbols of Σ over $\mathcal{D}_M$

Example: $\Sigma := (0, 1, +, -, >)$

- $\mathcal{D}$ is $\mathbb{N}$ or $\mathbb{Z}$
- 0 and 1 are the natural numbers *zero* and *one*
- $+ : (x, y) \mapsto x + y$, $- : (x, y) \mapsto x - y$
- $>: (x, y) \mapsto x > y$
- wff $w : \exists x.\forall y.\neg(x > y)$ (sentence)

Let $\mathcal{V}$ denote the set of variables.
Given an interpretation M, an **assignement** is a map $\sigma : \mathcal{V} \to \mathcal{D}_M$.
The assignement σ depends on the interpretation M.
The interpretation M associates

- Functions' symbols (f) of arity n to actual mathematical functions ($f_M : \mathcal{D}_M^n \to \mathcal{D}_M$)
- Terms to elements in $\mathcal{D}_M$
- Predicates' symbols (P) of arity n to subsets P_M in $\mathcal{D}_M^n$

Inductive Interpretation of wff

- $[\![P t_1 t_2]\!]_{M,\sigma} \triangleq ([\![t_1]\!]_\sigma, [\![t_2]\!]_\sigma) \in P_M$.
- $[\![\forall v.w]\!]_{M,\sigma} \triangleq [\![w[v \setminus m]]\!]_\sigma = 1$ for every m in the domain of M (m is a fresh variable not appearing in w).

Let M denote an interpretation, and $\sigma : \mathcal{V} \rightarrow \mathcal{D}_M$ an assignement.

- σ **satisfies** w w.r.t. M if and only if $[\![w]\!]_{M,\sigma} = 1$
- w is **satisfiable** w.r.t. M if there exist M, σ such that σ satisfies w w.r.t. M
- w is **unsatisfiable** if and only if:

$$\forall M. \forall \sigma. \ ([\![w]\!]_{M,\sigma} = 0) \quad .$$

Implication (w_i are wffs)

$w_1, \ldots, w_n \models w$ if and only if $\forall M, \sigma.\ (\llbracket \wedge_i w_i \rrbracket_{M,\sigma} = 1 \longrightarrow \llbracket w \rrbracket_{M,\sigma} = 1)$

Every interpretation and assignment that satisfy all w_i satisfy necessarily w

Definitions

- $\models w$: w is **valid**
- $w_1 \sim w_2$: $w_1 \models w_2$ and $w_2 \models w_1$ (equivalence)

Examples

- $\forall v_1.Pv_1 \models Pv_2$
- $\forall v_1.Pv_1 \models \exists v_2.Pv_2$
- $\exists v_1.\forall v_2.Qv_1v_2 \models \forall v_2.\exists v_1.Qv_1v_2$
- $\models \exists v_1(Pv_1 \longrightarrow \forall v_2.Pv_2)$
- $\forall v_1.\exists v_2.Qv_1v_2 \not\models \exists v_2.\forall v_1.Qv_1v_2$
- $Pv_1 \models \forall v_1.Pv_1$ (Depends on M)

Examples

- $\forall v_1.Pv_1 \models Pv_2$
- $\forall v_1.Pv_1 \models \exists v_2.Pv_2$
- $\exists v_1.\forall v_2.Qv_1v_2 \models \forall v_2.\exists v_1.Qv_1v_2$
- $\models \exists v_1(Pv_1 \longrightarrow \forall v_2.Pv_2)$
- $\forall v_1.\exists v_2.Qv_1v_2 \not\models \exists v_2.\forall v_1.Qv_1v_2$
- $Pv_1 \models \forall v_1.Pv_1$ (Depends on M)

Examples

- $\forall v_1.Pv_1 \models Pv_2$
- $\forall v_1.Pv_1 \models \exists v_2.Pv_2$
- $\exists v_1.\forall v_2.Qv_1v_2 \models \forall v_2.\exists v_1.Qv_1v_2$
- $\models \exists v_1(Pv_1 \longrightarrow \forall v_2.Pv_2)$
- $\forall v_1.\exists v_2.Qv_1v_2 \not\models \exists v_2.\forall v_1.Qv_1v_2$
- $Pv_1 \models \forall v_1.Pv_1$ (Depends on M)

Examples

- $\forall v_1.Pv_1 \models Pv_2$
- $\forall v_1.Pv_1 \models \exists v_2.Pv_2$
- $\exists v_1.\forall v_2.Qv_1v_2 \models \forall v_2.\exists v_1.Qv_1v_2$
- $\models \exists v_1(Pv_1 \longrightarrow \forall v_2.Pv_2)$
- $\forall v_1.\exists v_2.Qv_1v_2 \not\models \exists v_2.\forall v_1.Qv_1v_2$
- $Pv_1 \models \forall v_1.Pv_1$ (Depends on M)

Examples

- $\forall v_1.Pv_1 \models Pv_2$
- $\forall v_1.Pv_1 \models \exists v_2.Pv_2$
- $\exists v_1.\forall v_2.Qv_1v_2 \models \forall v_2.\exists v_1.Qv_1v_2$
- $\models \exists v_1(Pv_1 \longrightarrow \forall v_2.Pv_2)$
- $\forall v_1.\exists v_2.Qv_1v_2 \not\models \exists v_2.\forall v_1.Qv_1v_2$
- $Pv_1 \models \forall v_1.Pv_1$ (Depends on M)

Examples

- $\forall v_1.Pv_1 \models Pv_2$
- $\forall v_1.Pv_1 \models \exists v_2.Pv_2$
- $\exists v_1.\forall v_2.Qv_1v_2 \models \forall v_2.\exists v_1.Qv_1v_2$
- $\models \exists v_1(Pv_1 \longrightarrow \forall v_2.Pv_2)$
- $\forall v_1.\exists v_2.Qv_1v_2 \not\models \exists v_2.\forall v_1.Qv_1v_2$
- $Pv_1 \models \forall v_1.Pv_1$ (Depends on M)

Quantifier free formula: $(x \leq 0 \vee x + y \leq 0) \wedge y \geq 1 \wedge x \geq 1$
Translated into a CNF: $(a \vee b) \wedge c \wedge d$

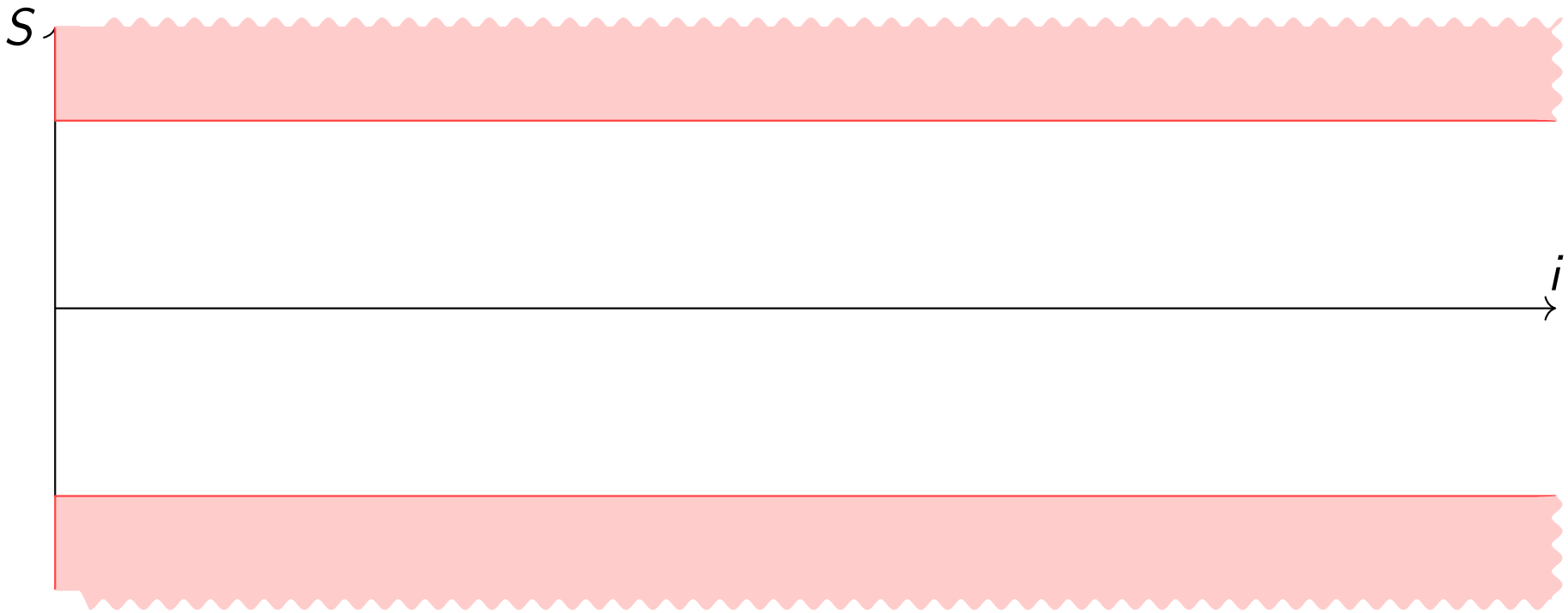
SAT gives $(a, b, c, d) = (1, 0, 1, 1)$
But $x \leq 0 \wedge x \geq 1$ is a contradiction:
Learn $\bar{a} \vee \bar{d}$

SAT gives $(a, b, c, d) = (0, 1, 1, 1)$
But $x + y \leq 0 \wedge y \geq 1 \wedge x \geq 1$ is a contradiction:
Learn $\bar{b} \vee \bar{c} \vee \bar{d}$
The problem is UNSAT.

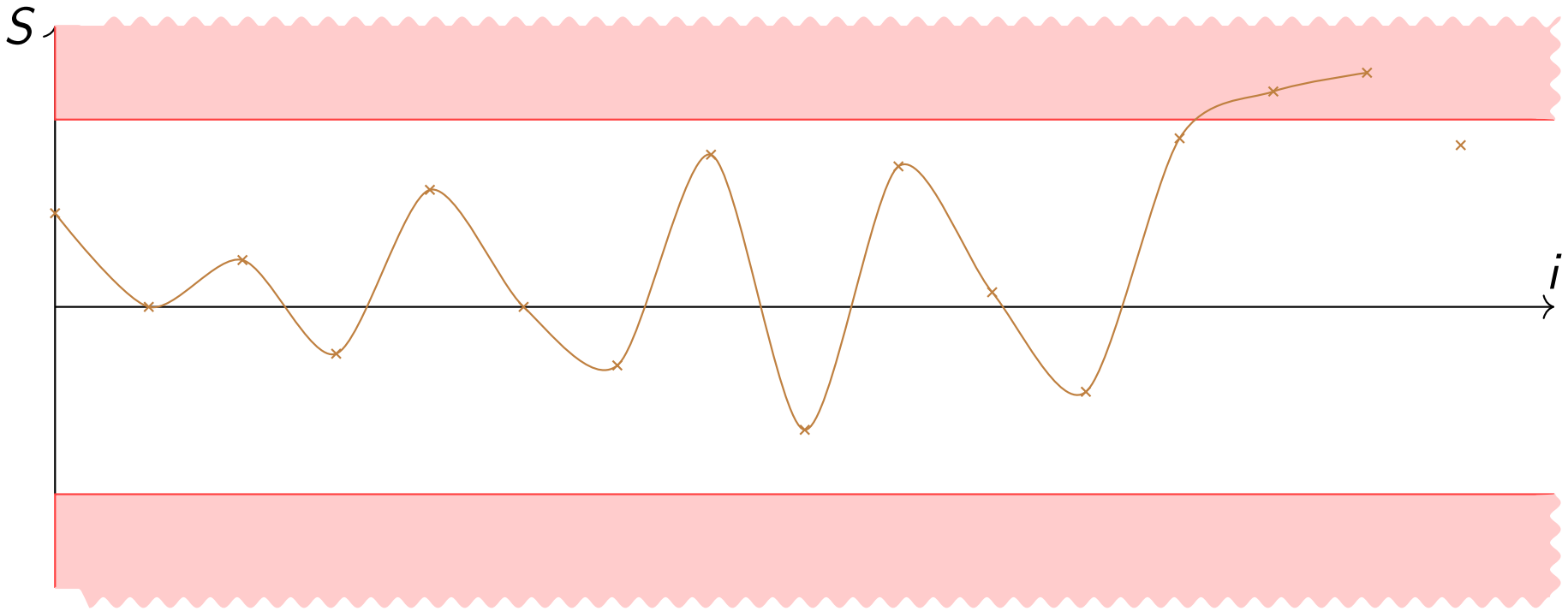
- Linear arithmetic can be decided by exact precision Simplex (upcoming courses)

- There are other generic techniques based on **abstract interpretation**

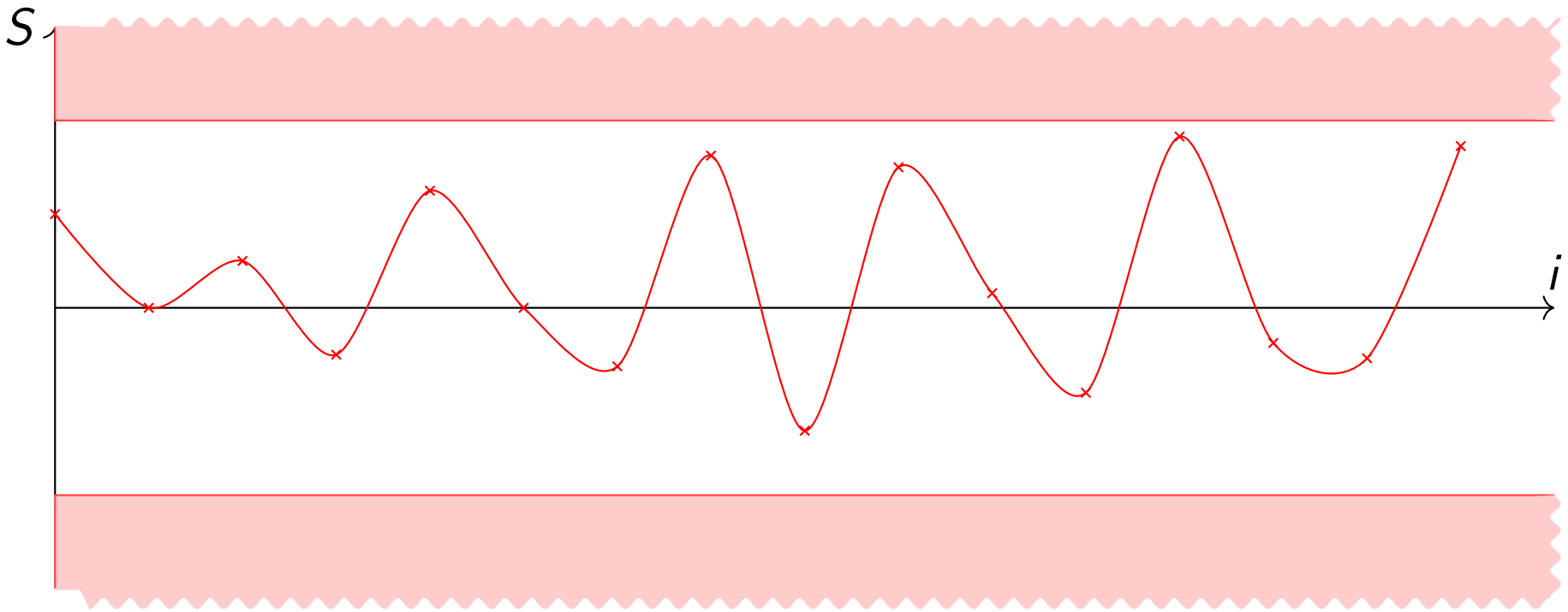
Abstract Interpretation : Intuitions



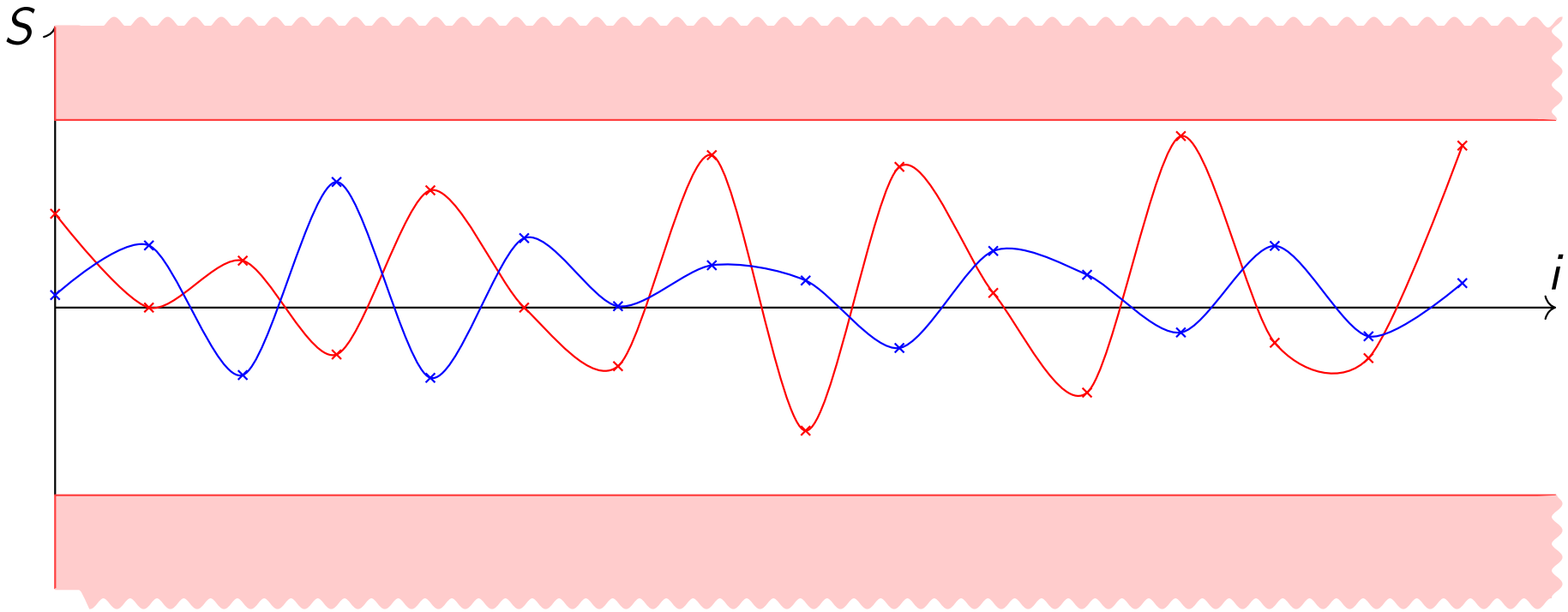
Abstract Interpretation : Intuitions



Abstract Interpretation : Intuitions

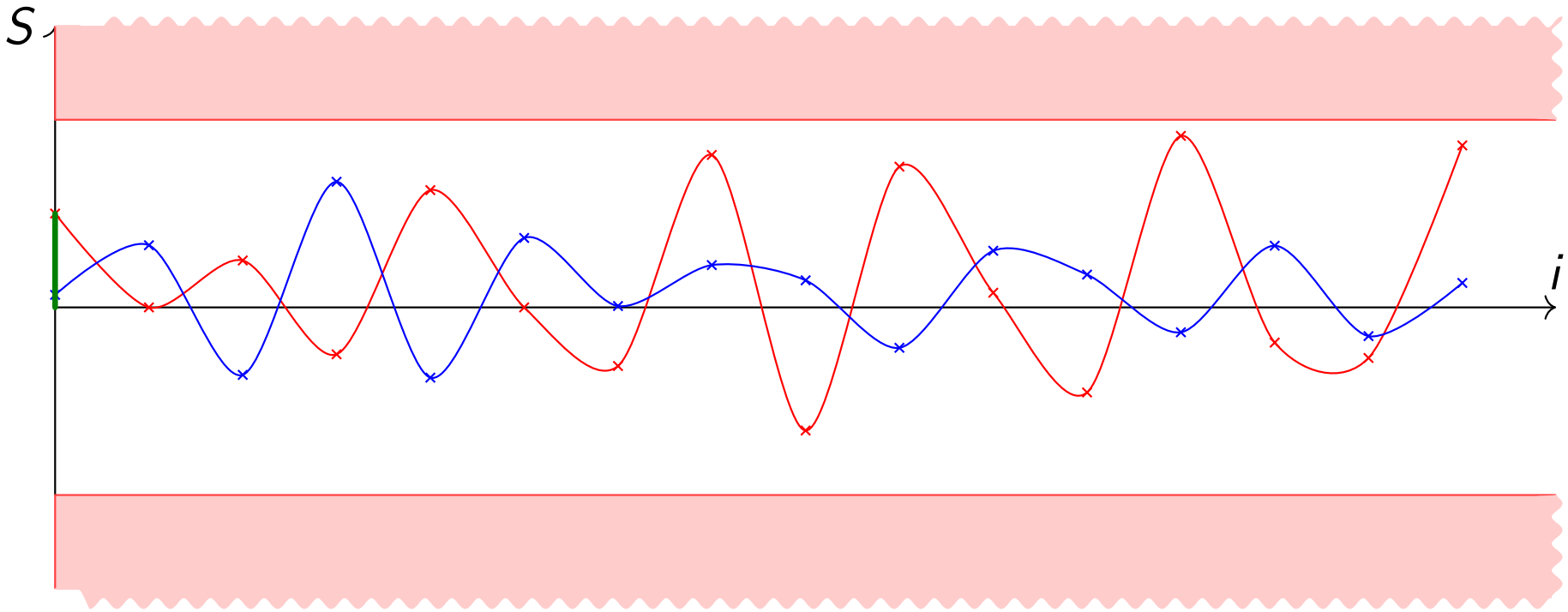


Abstract Interpretation : Intuitions

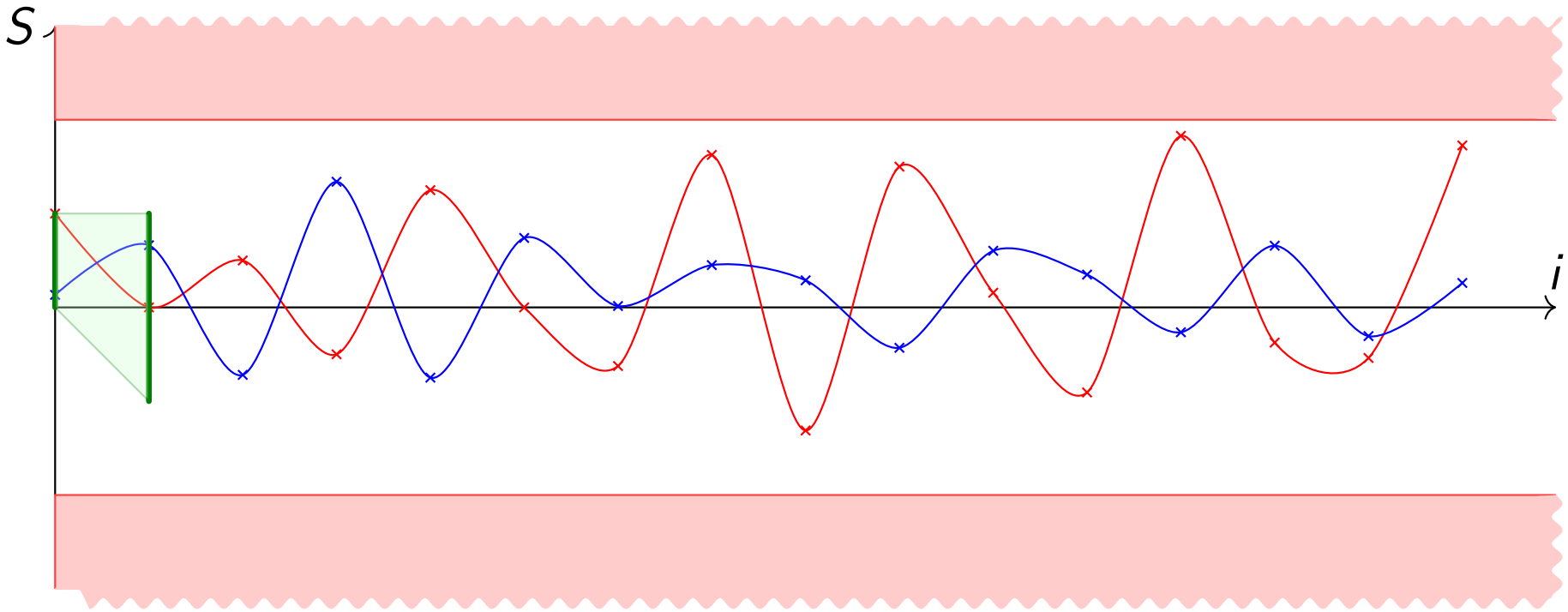


- What about the missed bugs ? are they severe ?

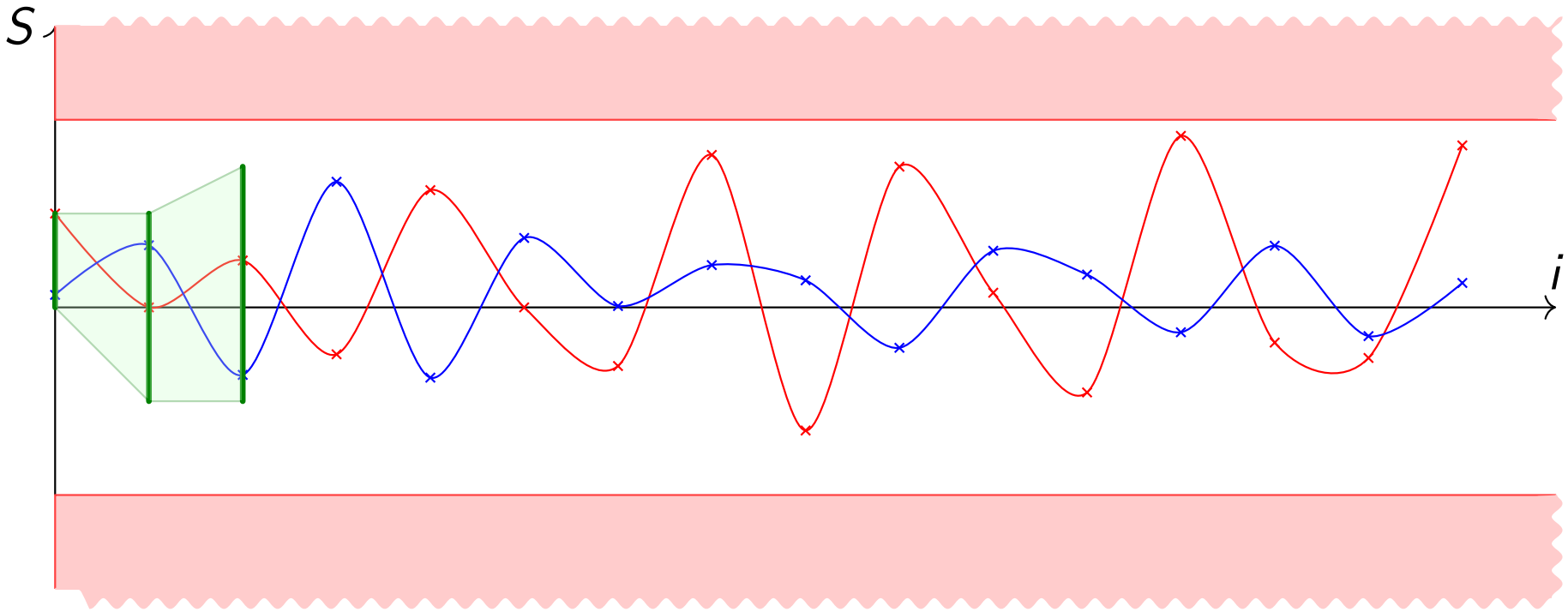
Abstract Interpretation : Intuitions



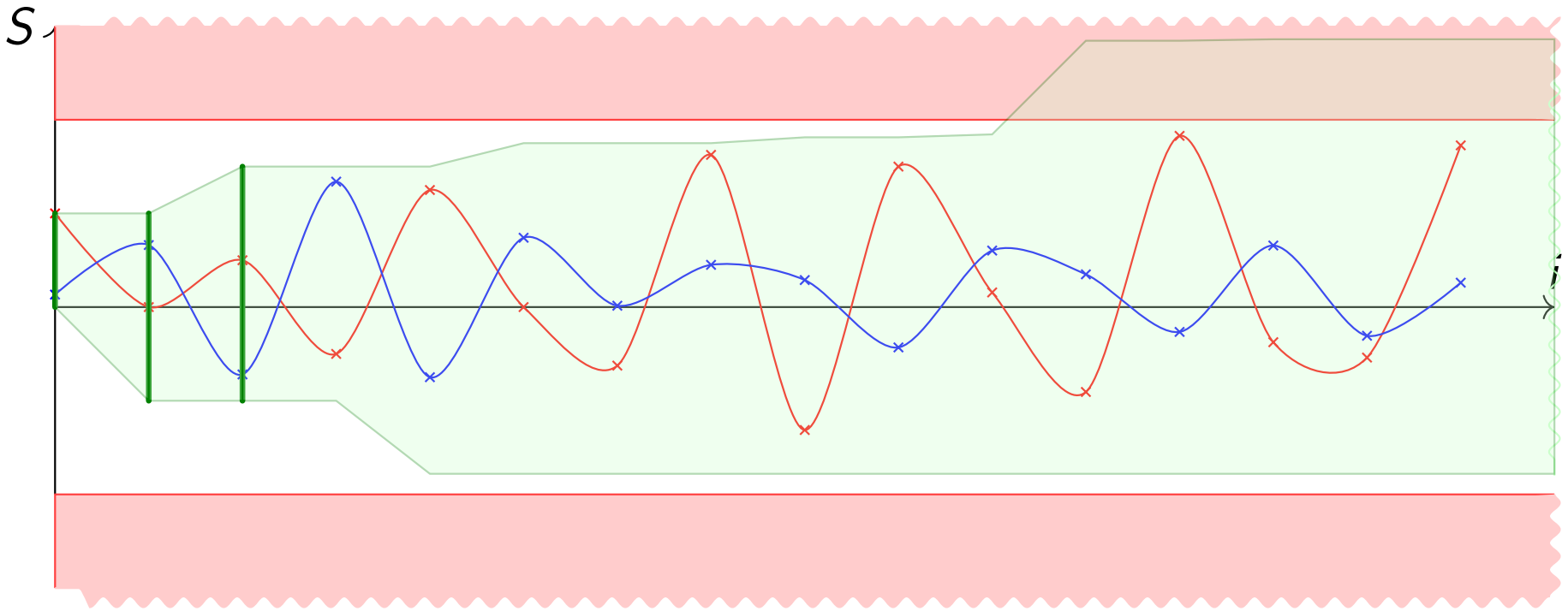
Abstract Interpretation : Intuitions



Abstract Interpretation : Intuitions

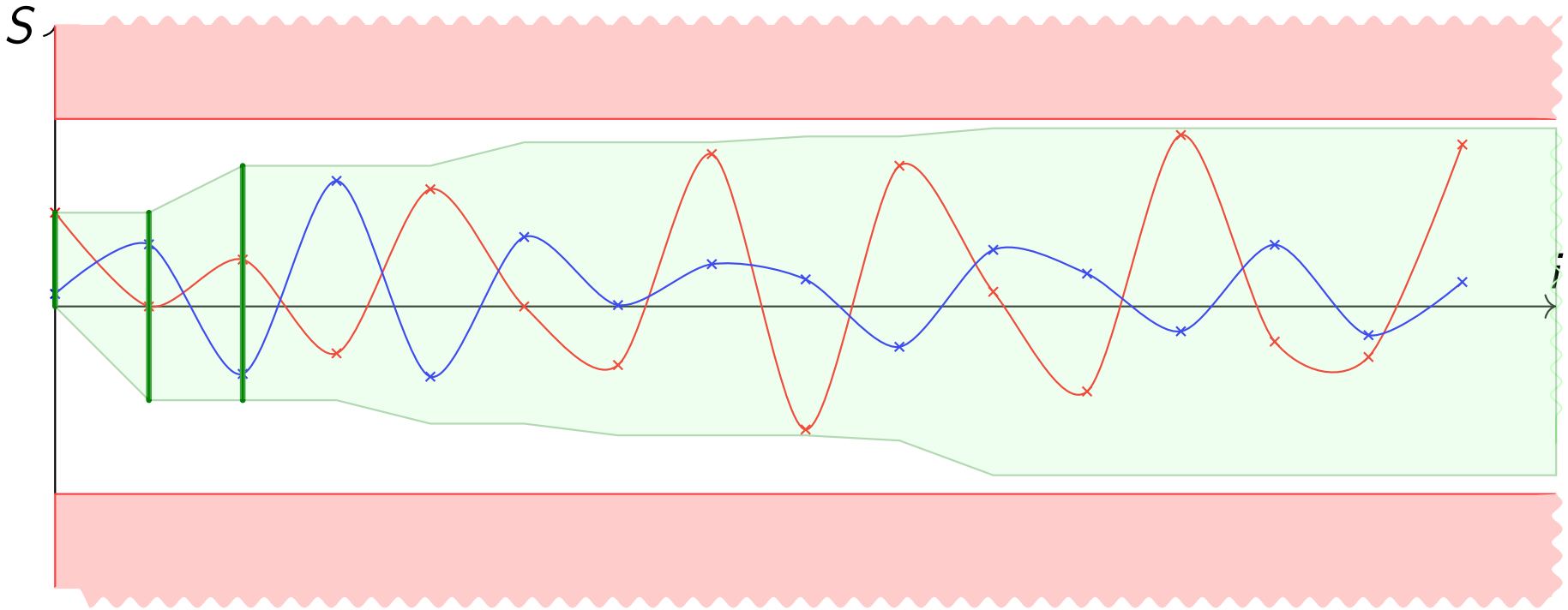


Abstract Interpretation : Intuitions



- Over-approximation may lead to **false alarms**.

Abstract Interpretation : Intuitions



- Accurate over-approximation gives a safety **proof**.

Examples

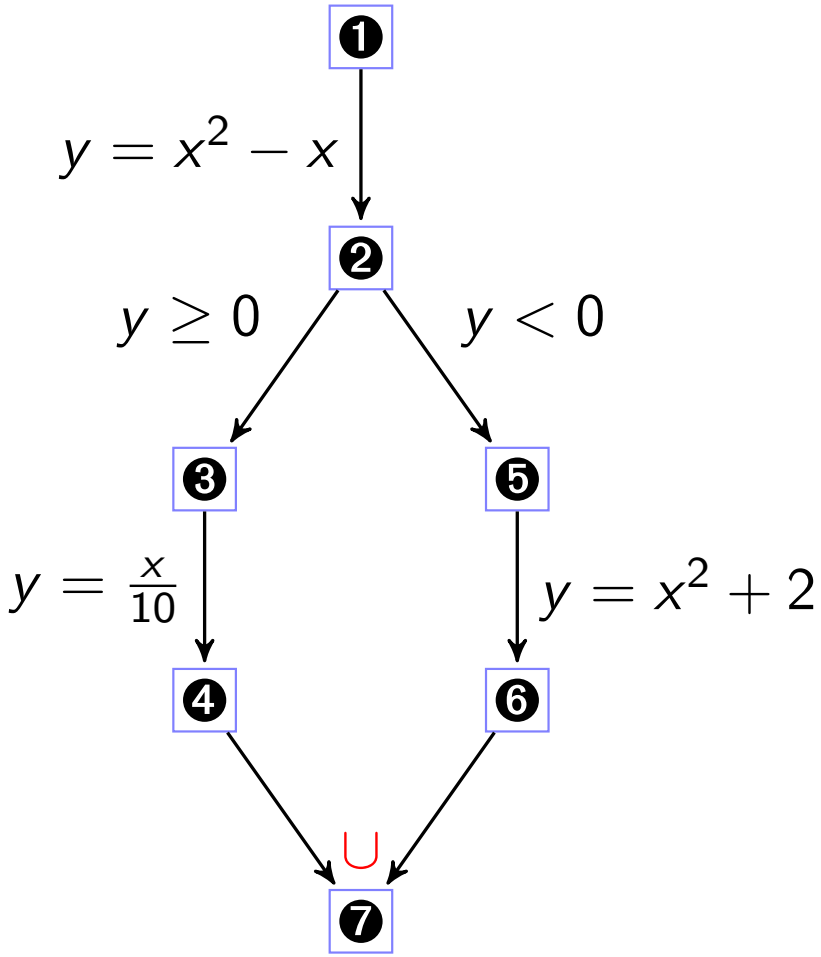
- 1982, The Vancouver stock exchange: after 22 months the index had fallen to $524,811$ instead of $1098,811$
- 1985, Therac 25 (radiation therapy machine) : 5 patients killed (overdoses of radiation)
- 1991, The Patriot Missile: 28 soldiers killed
- 1996, Ariane 5: more than 1 billion $ gone in 40 seconds

E. Dijkstra (1972)

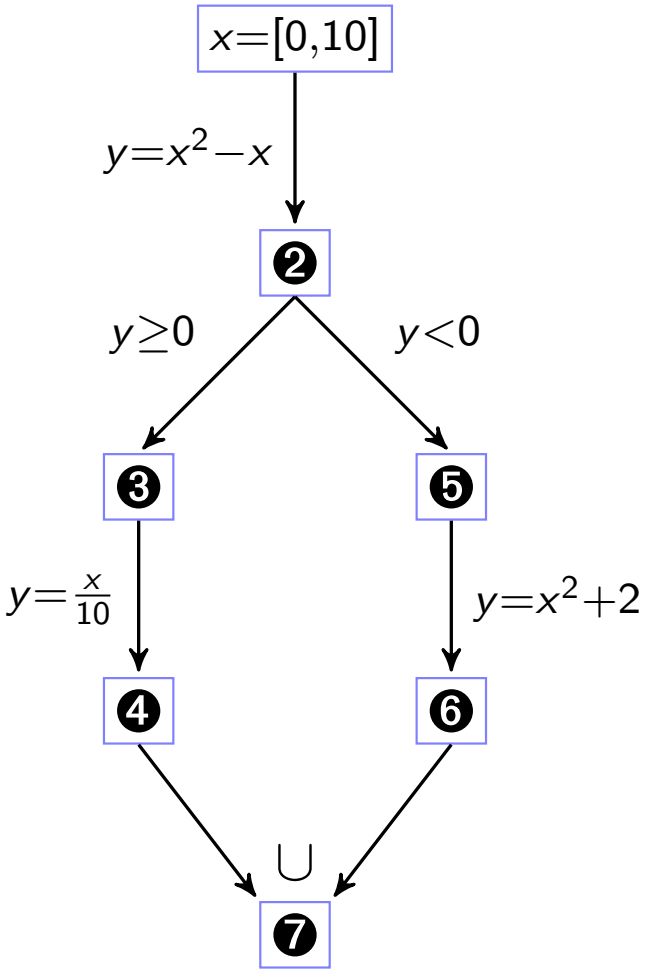
Program testing can be used to show the presence of bugs, but never to show their absence!

Detailed example

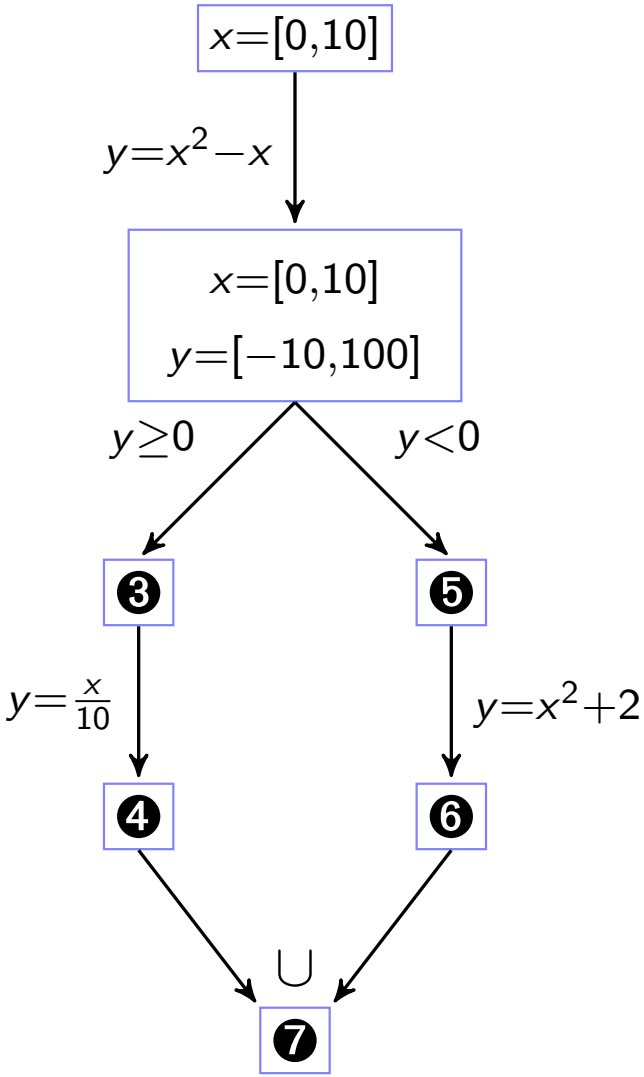
```
begin
x = [0,10]; ❶
y = x*x - x ❷
if (y >= 0)  ❸ then
y = x / 10; ❹
else  ❺
  y = x*x + 2; ❻
done; ❼
end
```



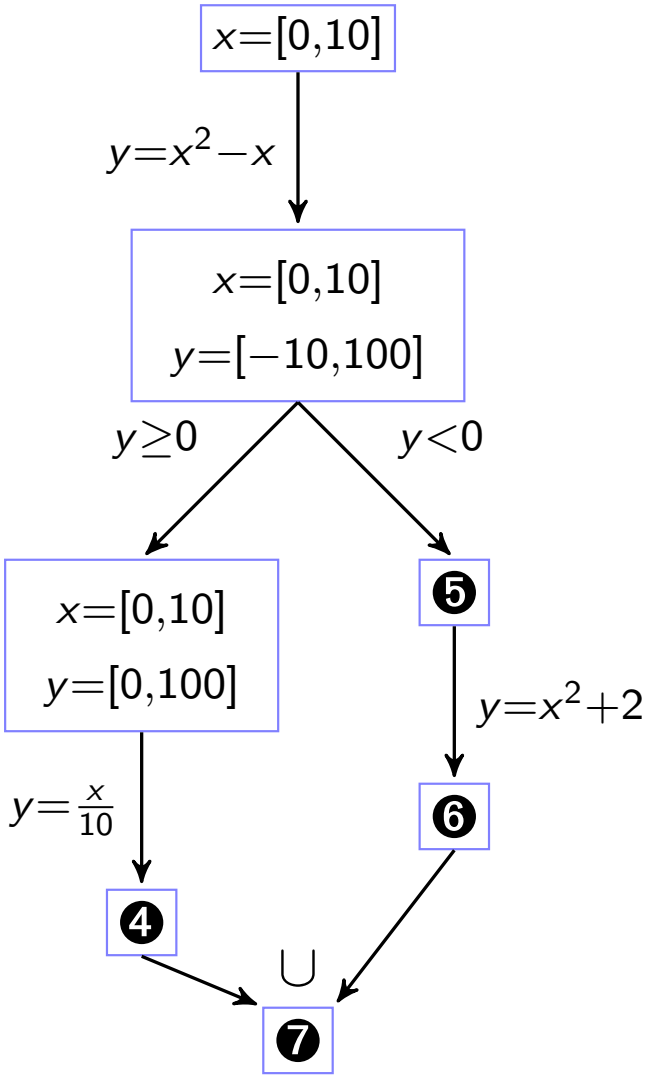
Forward Propagation



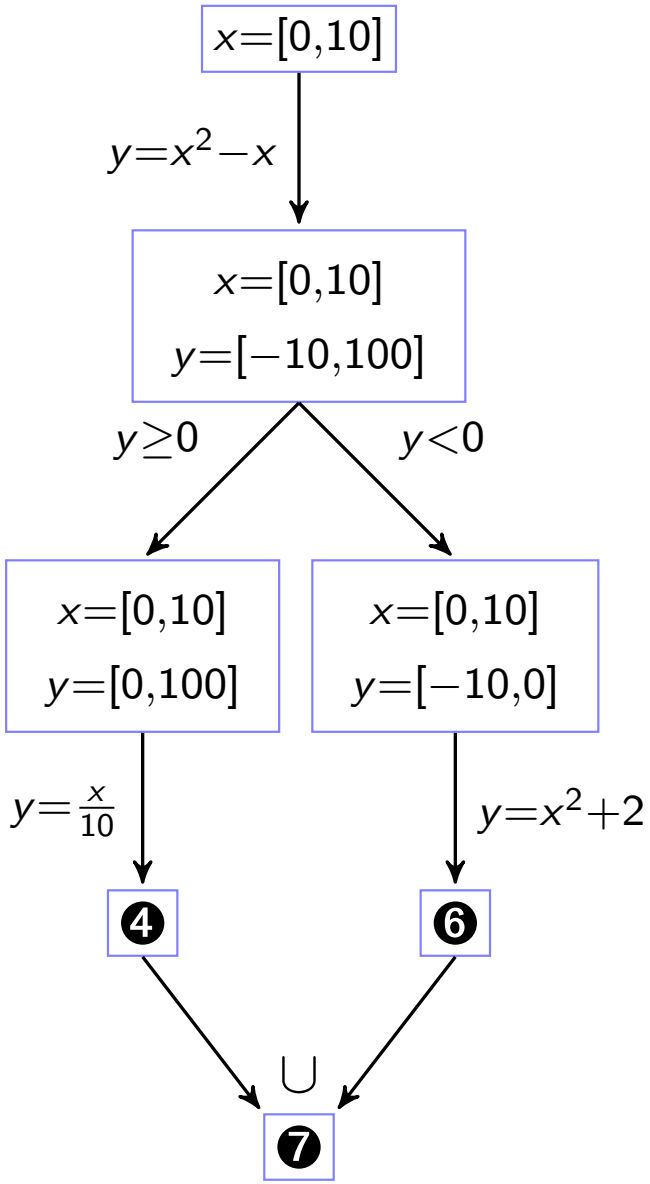
Forward Propagation



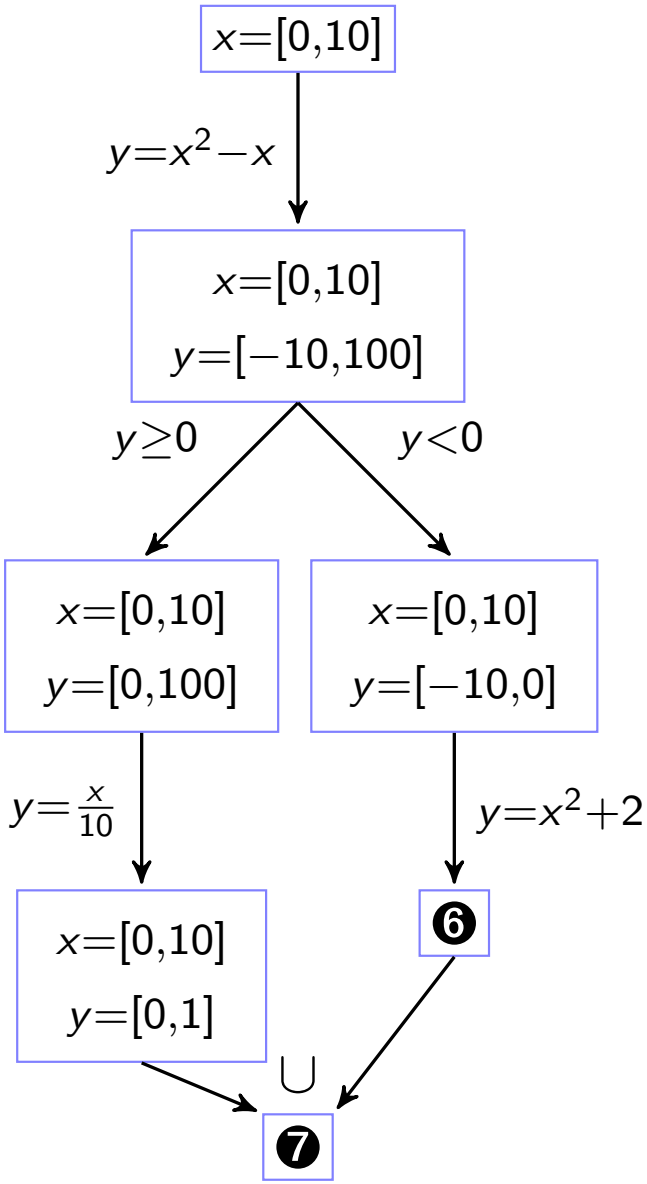
Forward Propagation



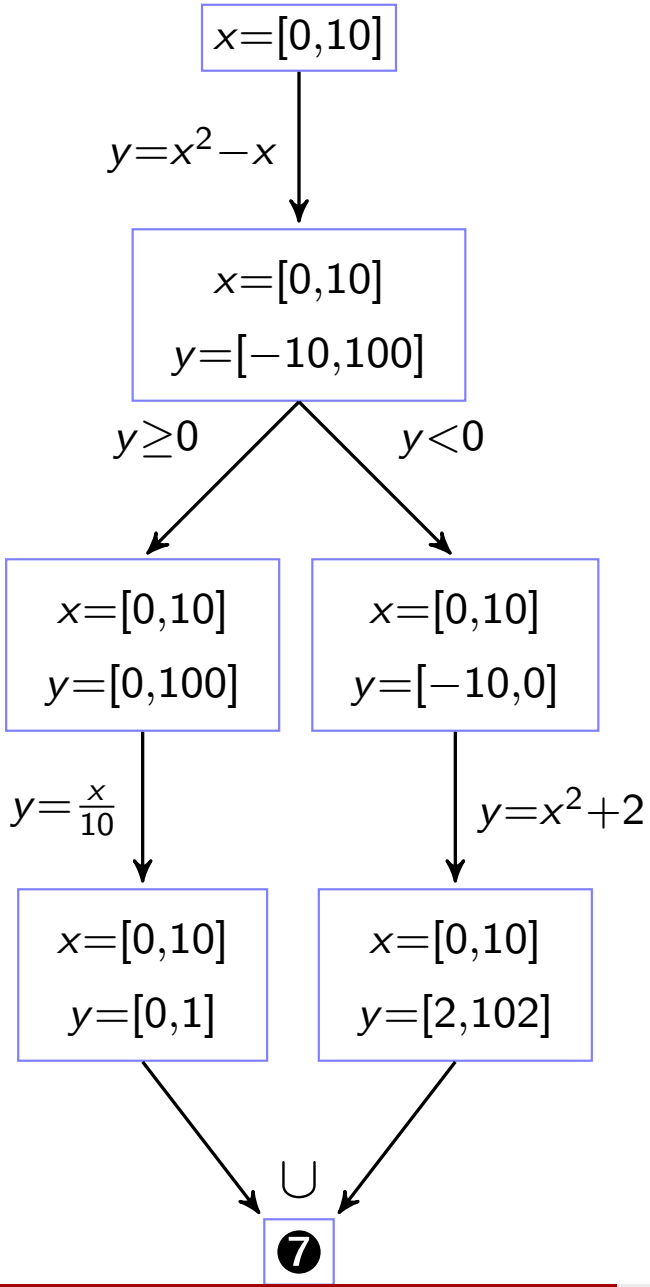
Forward Propagation



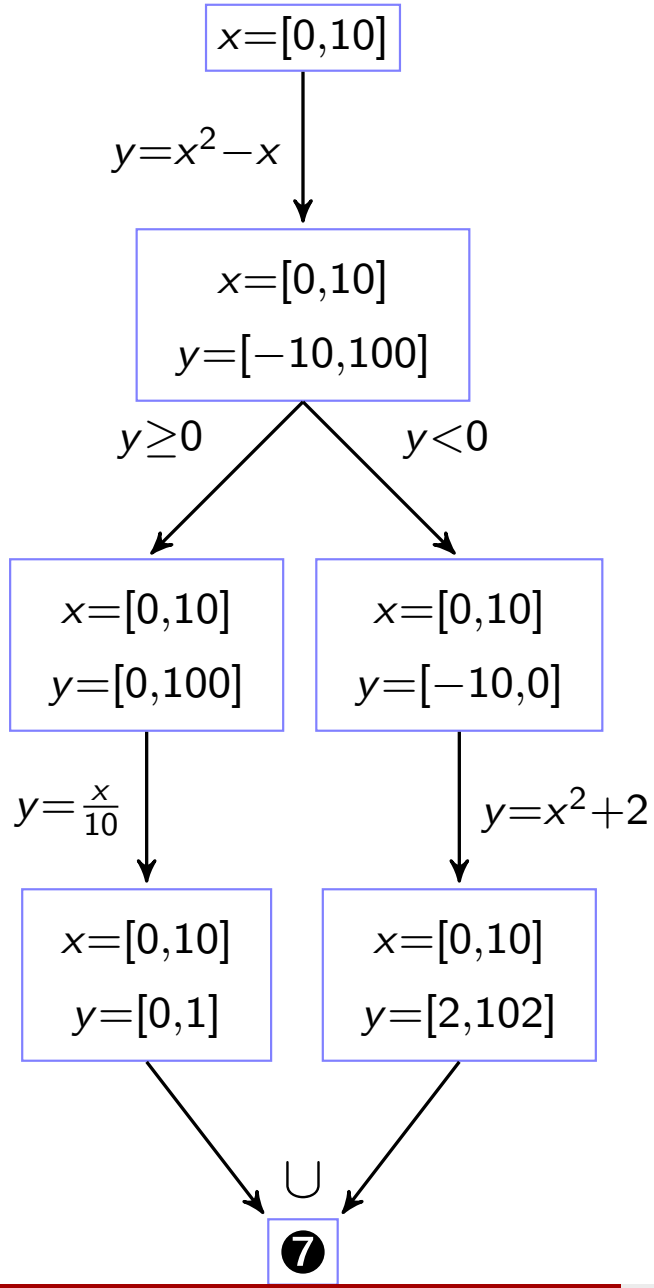
Forward Propagation

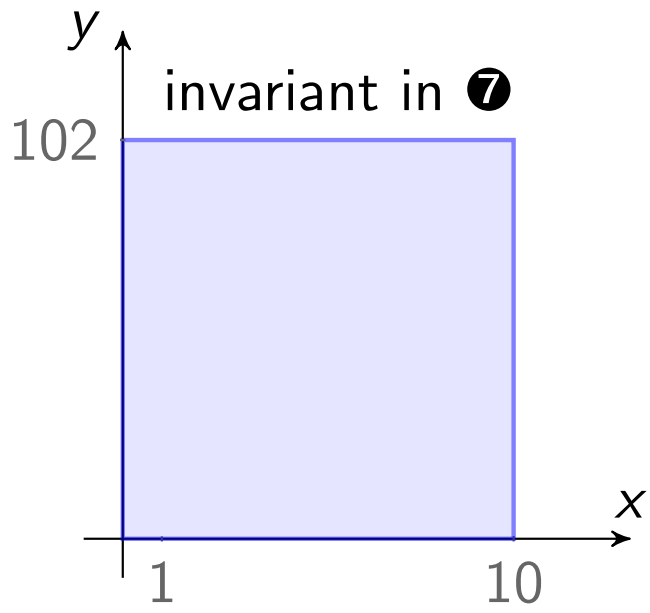


Forward Propagation

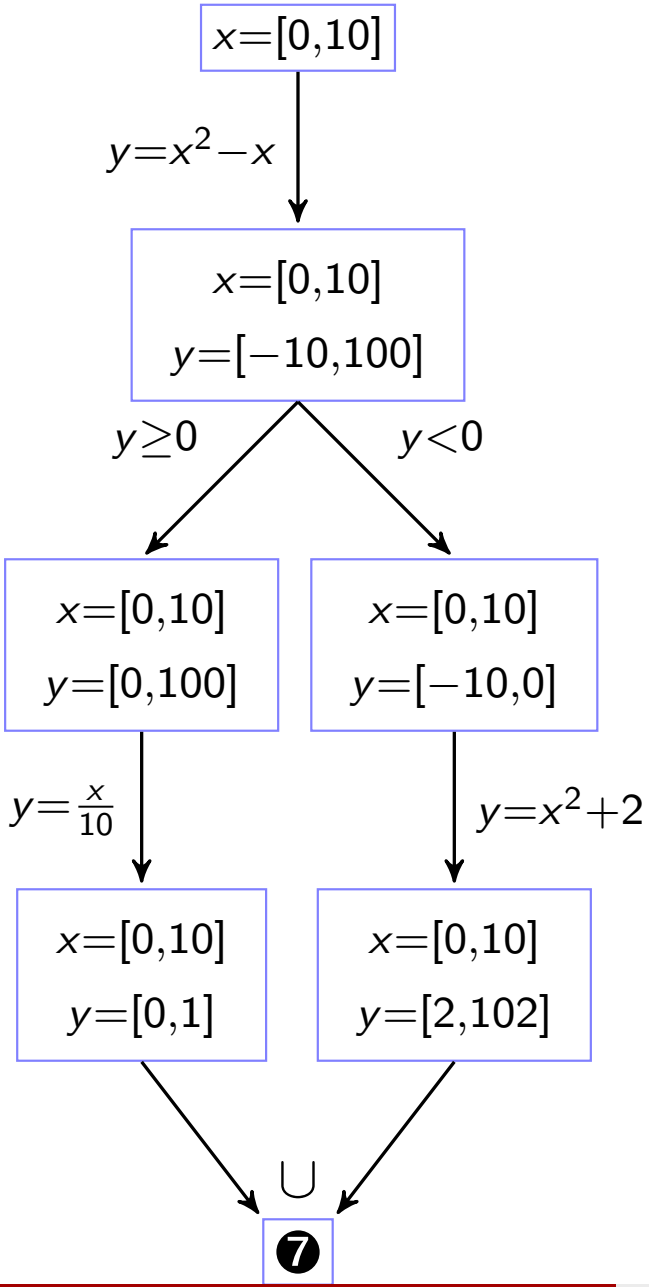


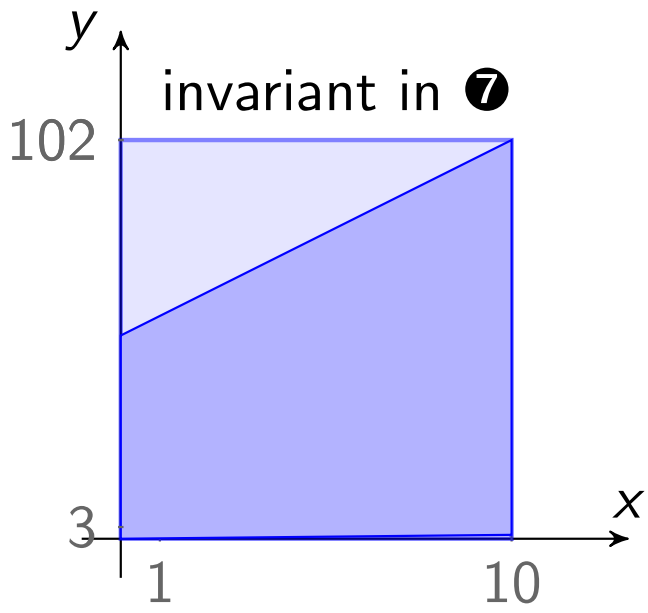
Forward Propagation

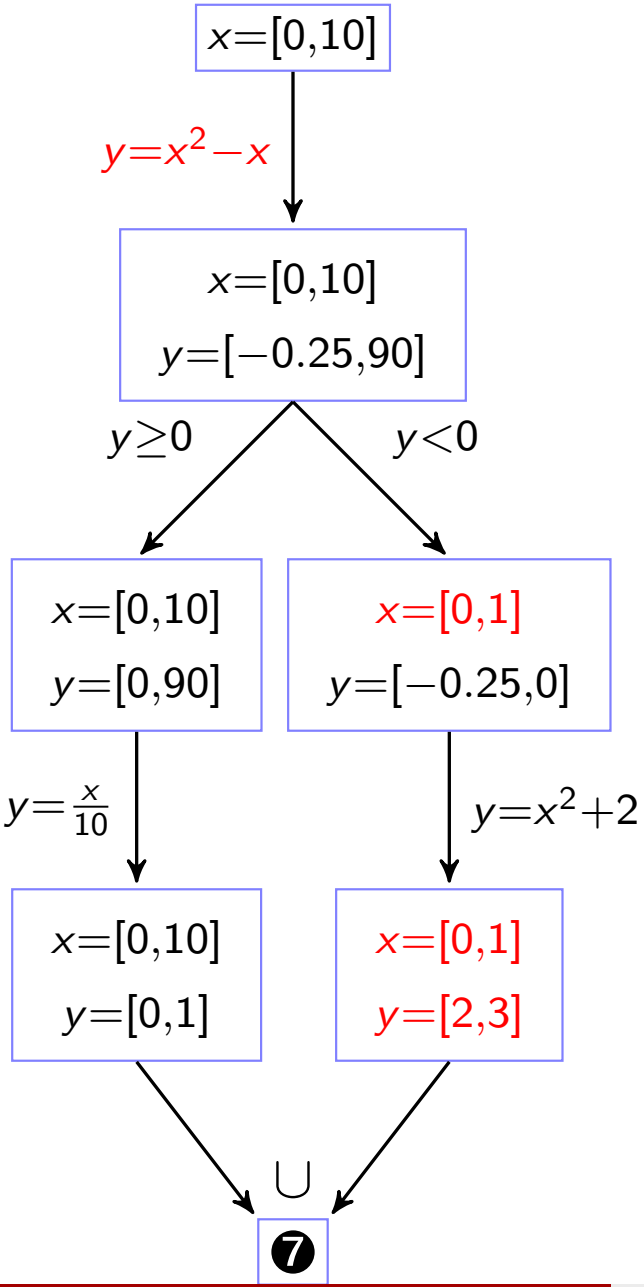


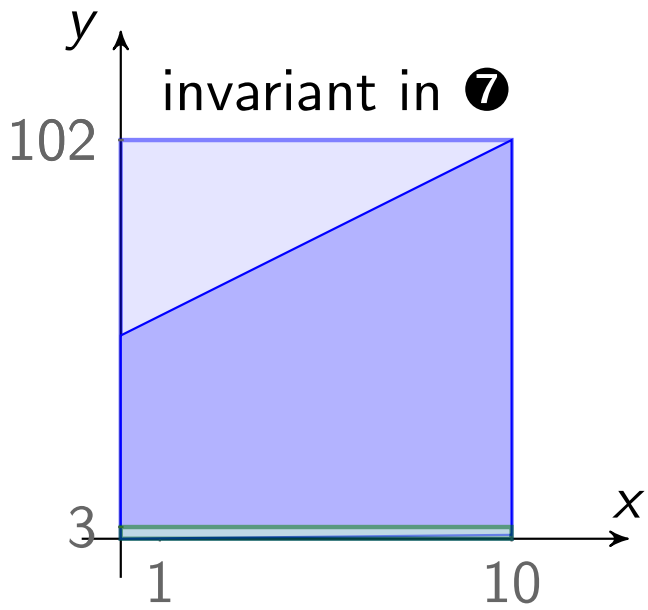


Forward Propagation

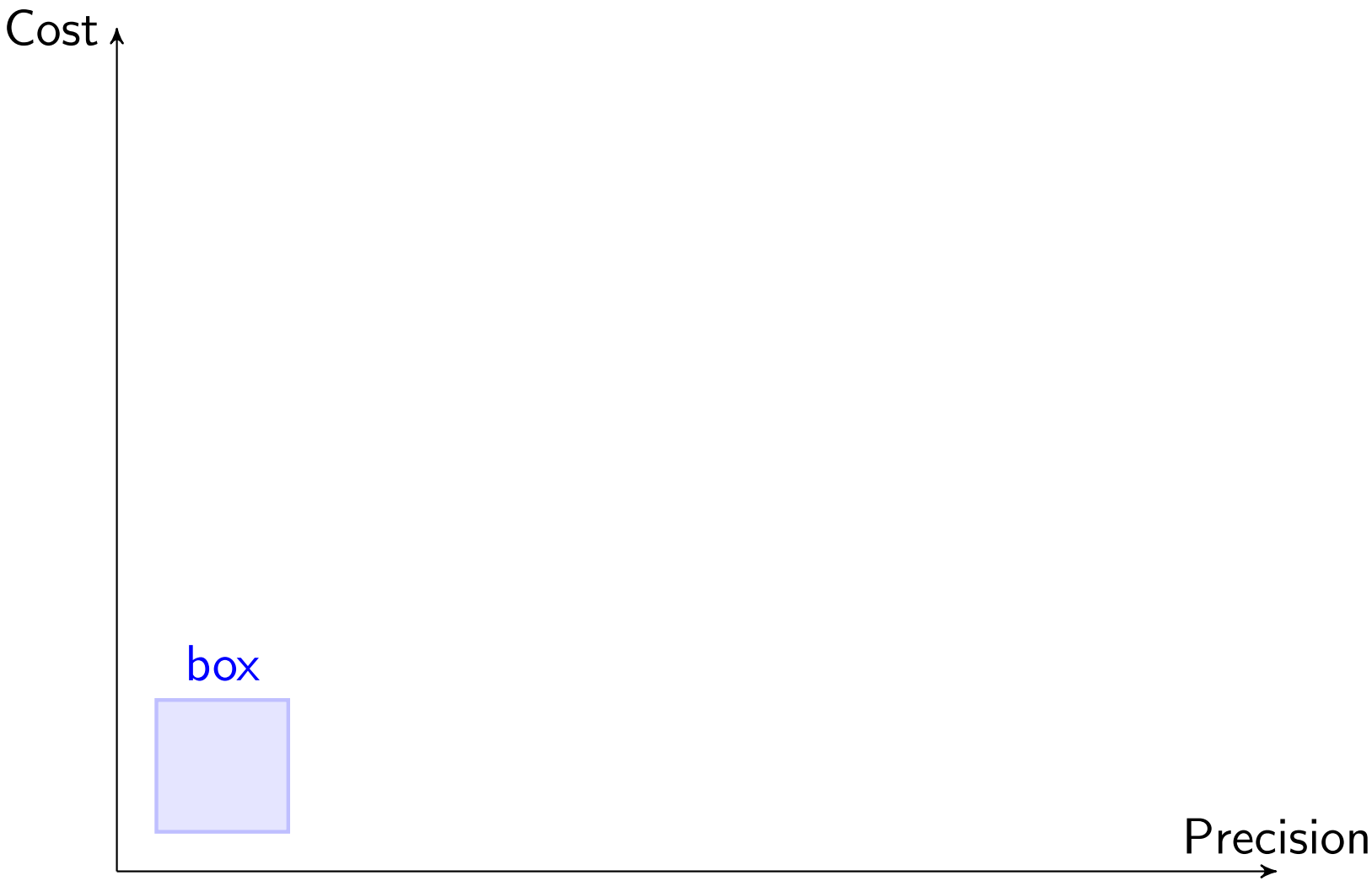




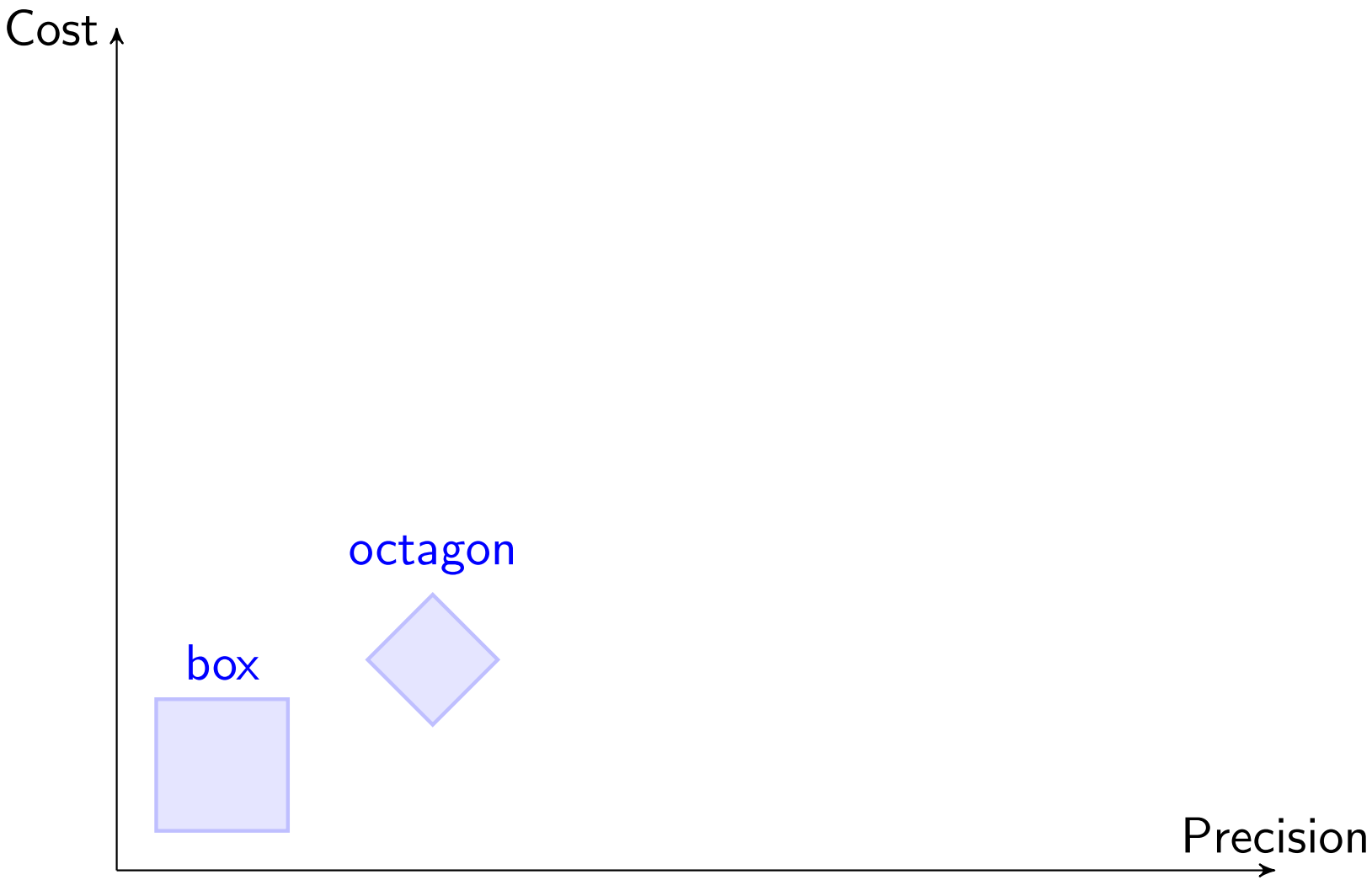

x=[0,10]
y=x²−x
x=[0,10]
y=[−0.25,90]
y≥0
y<0
x=[0,10]
y=[0,90]
x=[0,1]
y=[−0.25,0]
y=x/10
y=x²+2
x=[0,10]
y=[0,1]
x=[0,1]
y=[2,3]
∪
❼


y
invariant in ❼
102
3
1
10
x

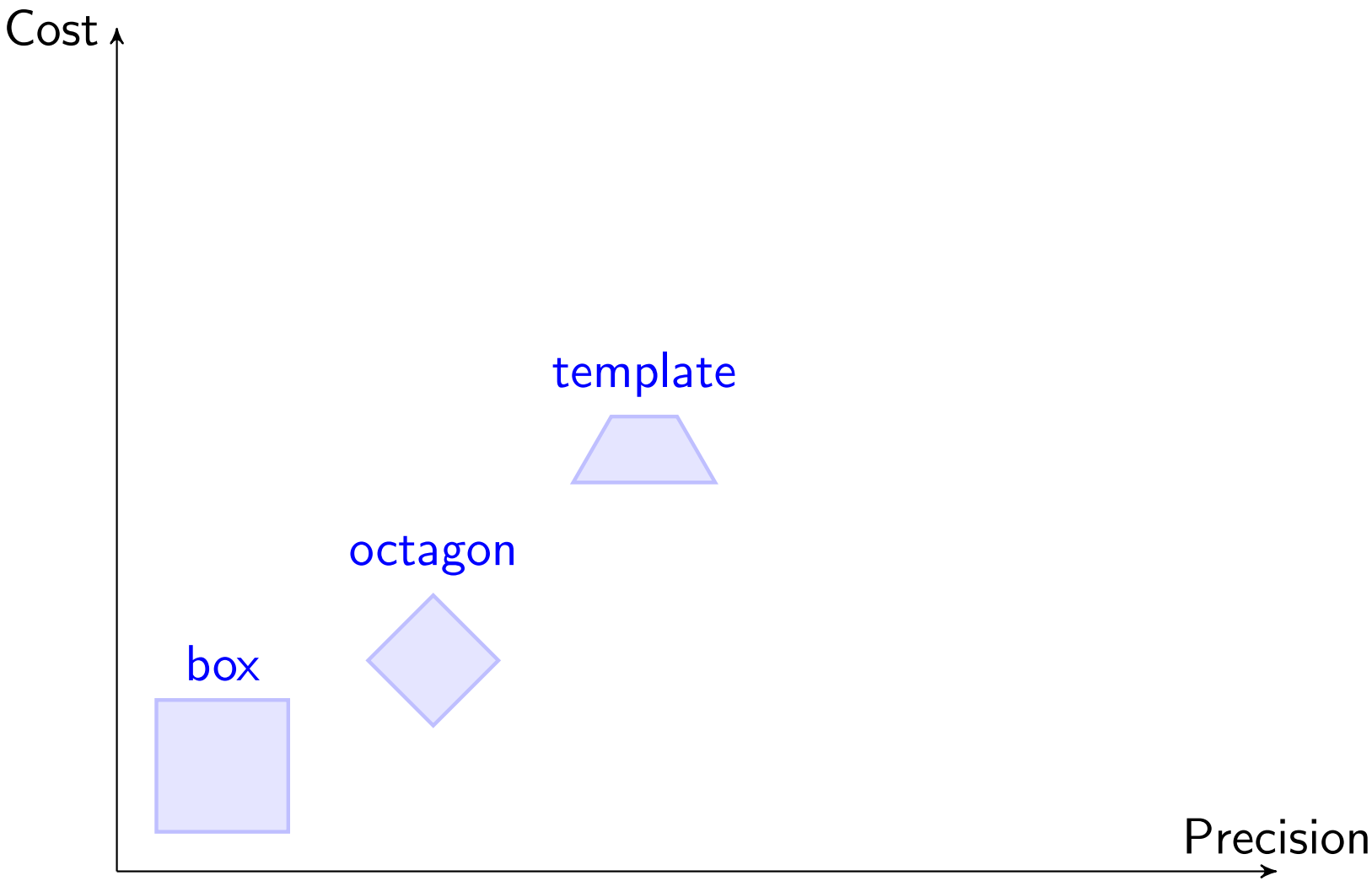
Precision Cost Trade-off



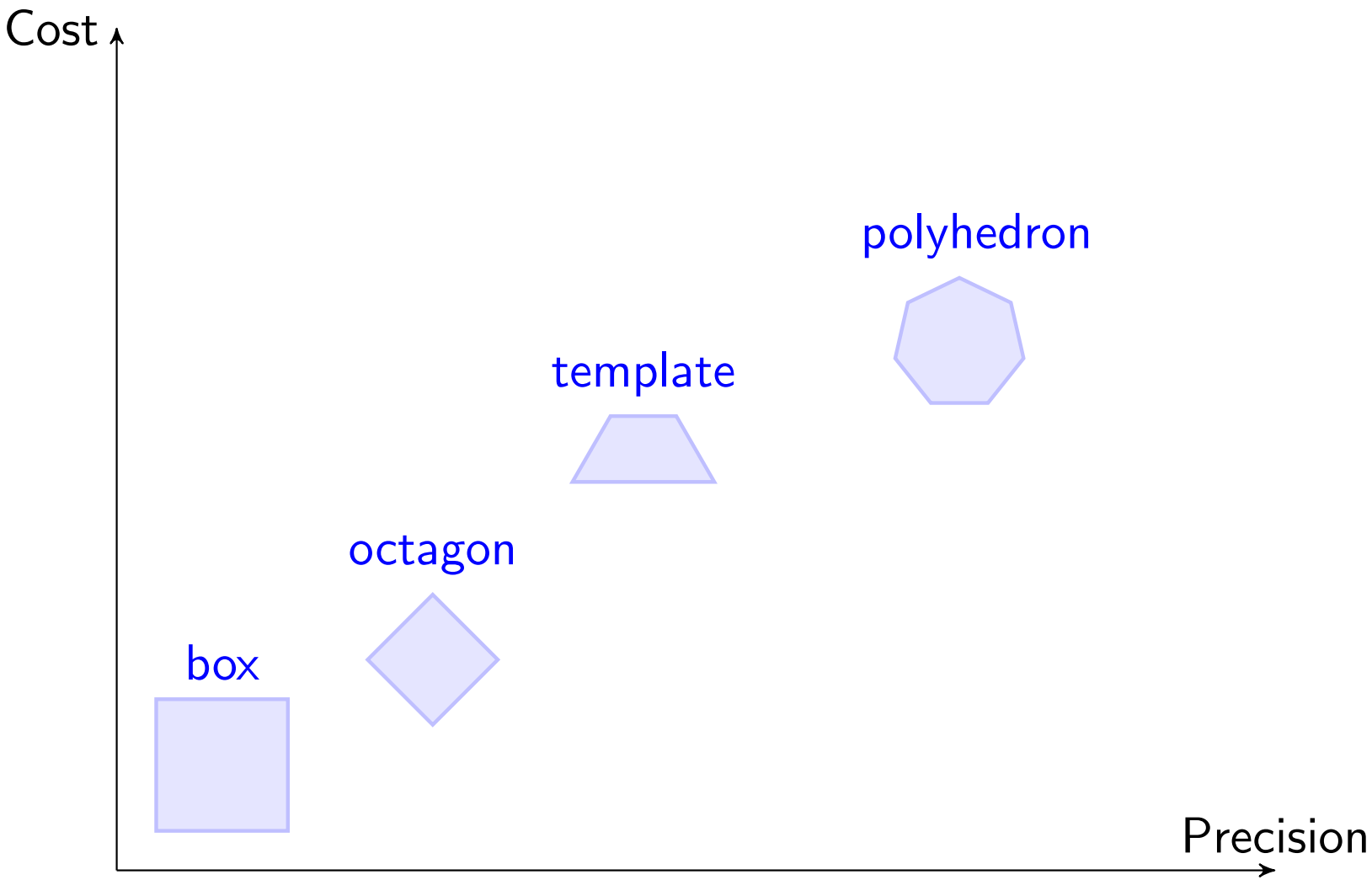
Precision Cost Trade-off



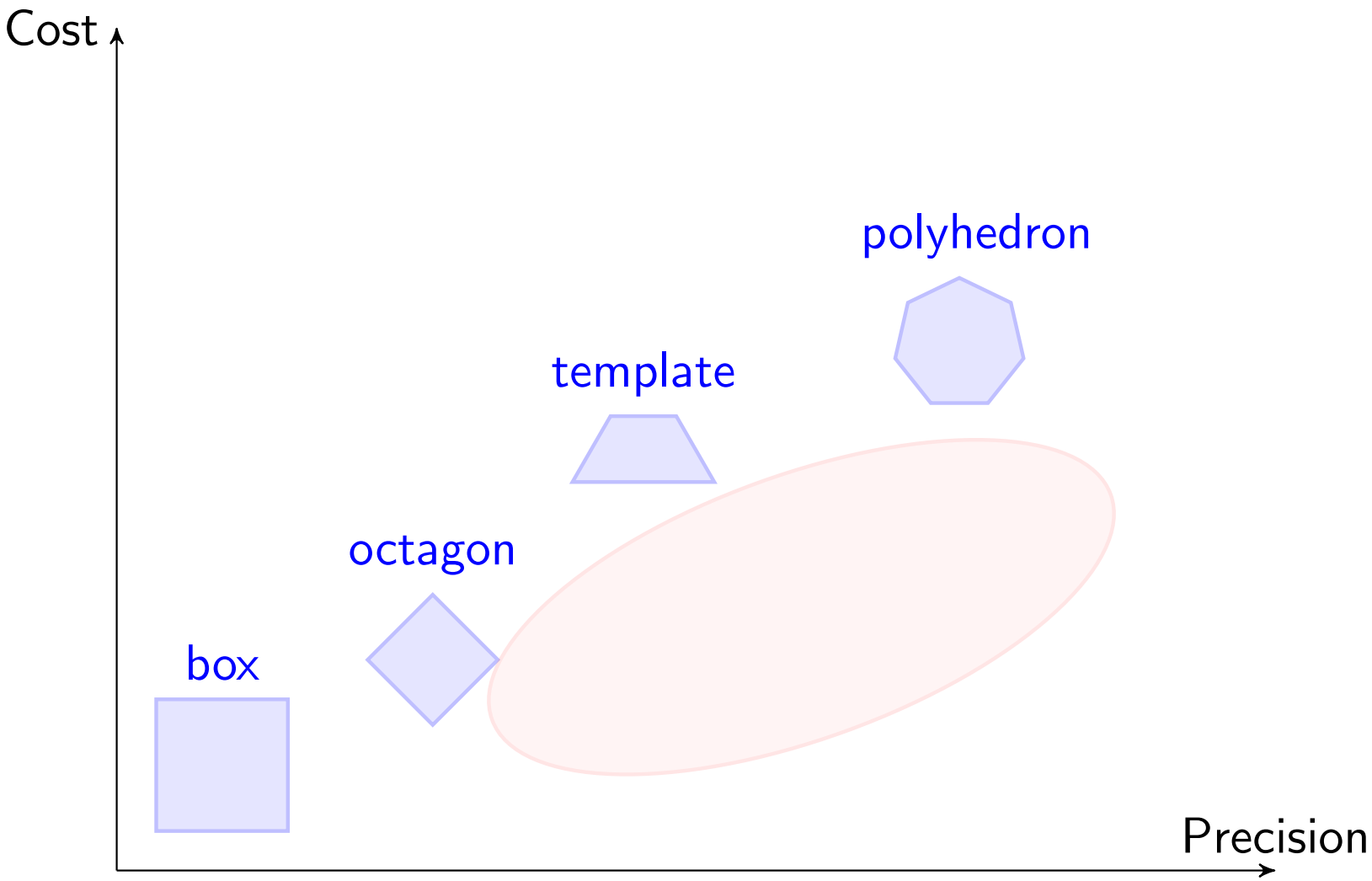
Precision Cost Trade-off



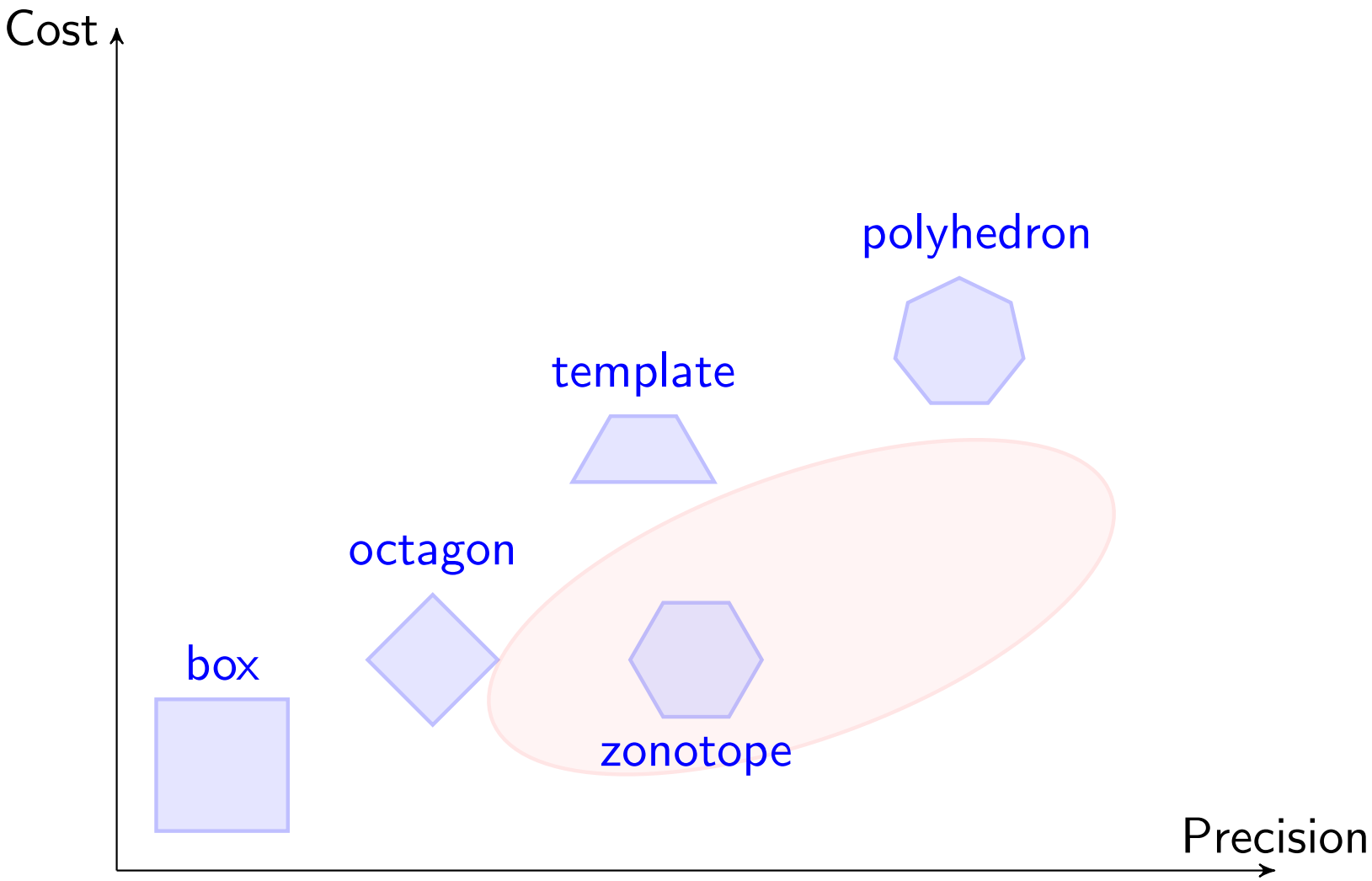
Precision Cost Trade-off



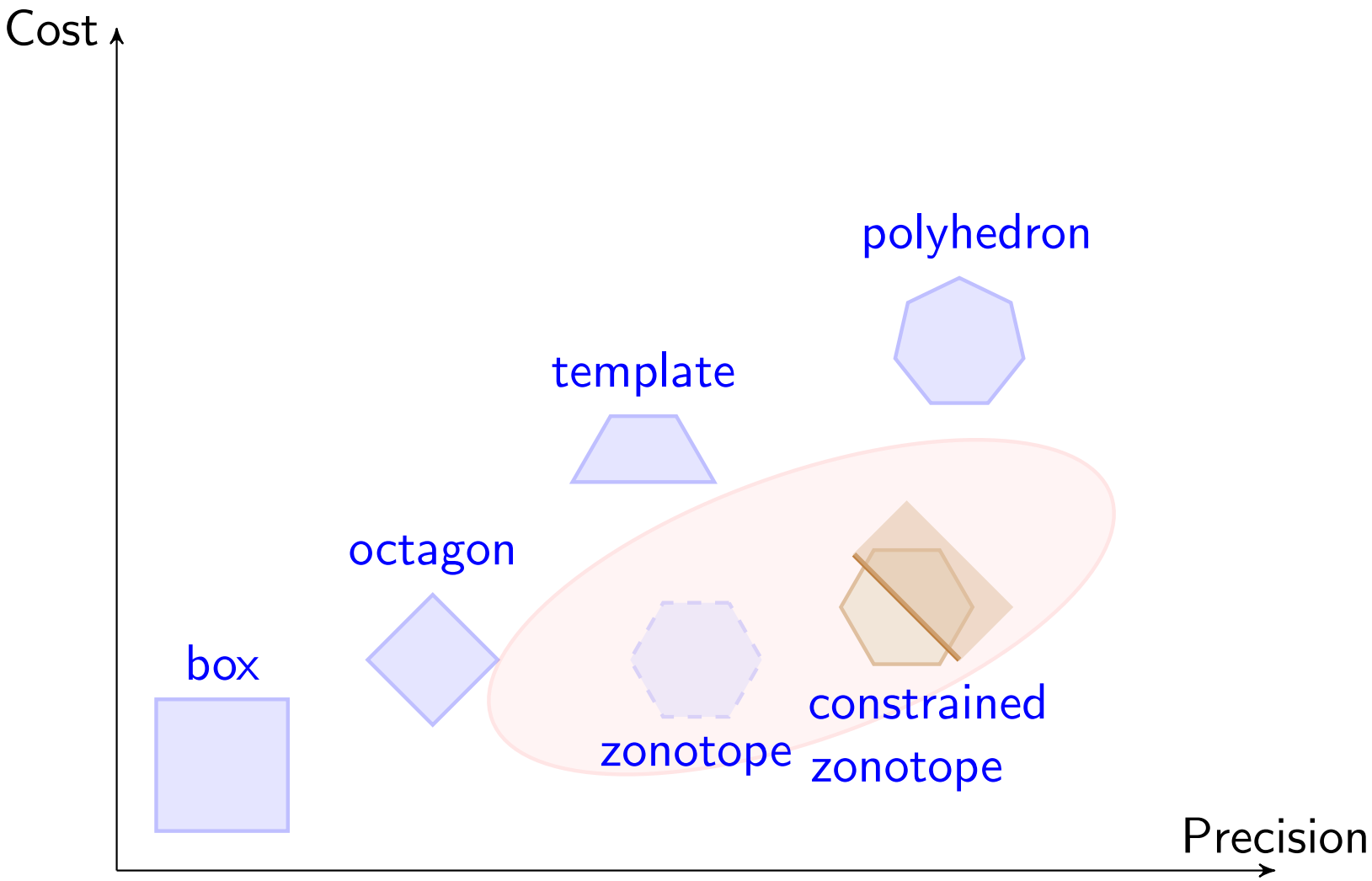
Precision Cost Trade-off



Precision Cost Trade-off



Precision Cost Trade-off



Outlines

1 First Order Logic

2 Static Analysis-based Abstract Interpretation

Formal Verification Approaches

- Hoare 1969: wrap the code of interest with preconditions and postconditions, then prove that postconditions are met
- Clarke, Emerson et Sifakis 1974: model checking
- Cousot(s) 1977: Abstract Interpretation

Properties of Interest

- run time errors: overflow, division by zero, square root of negatives, etc.
- robustness and stability of algorithms: linear and non linear recursive schemes, filters, etc.

Formal Verification Approaches

- Hoare 1969: wrap the code of interest with preconditions and postconditions, then prove that postconditions are met
- Clarke, Emerson et Sifakis 1974: model checking
- Cousot(s) 1977: Abstract Interpretation

Properties of Interest

- run time errors: overflow, division by zero, square root of negatives, etc.
- robustness and stability of algorithms: linear and non linear recursive schemes, filters, etc.

Formal Verification Approaches

- Hoare 1969: wrap the code of interest with preconditions and postconditions, then prove that postconditions are met
- Clarke, Emerson et Sifakis 1974: model checking
- Cousot(s) 1977: Abstract Interpretation

Properties of Interest

- run time errors: overflow, division by zero, square root of negatives, etc.
- robustness and stability of algorithms: linear and non linear recursive schemes, filters, etc.

Formal Verification Approaches

- Hoare 1969: wrap the code of interest with preconditions and postconditions, then prove that postconditions are met
- Clarke, Emerson et Sifakis 1974: model checking
- Cousot(s) 1977: Abstract Interpretation

Properties of Interest

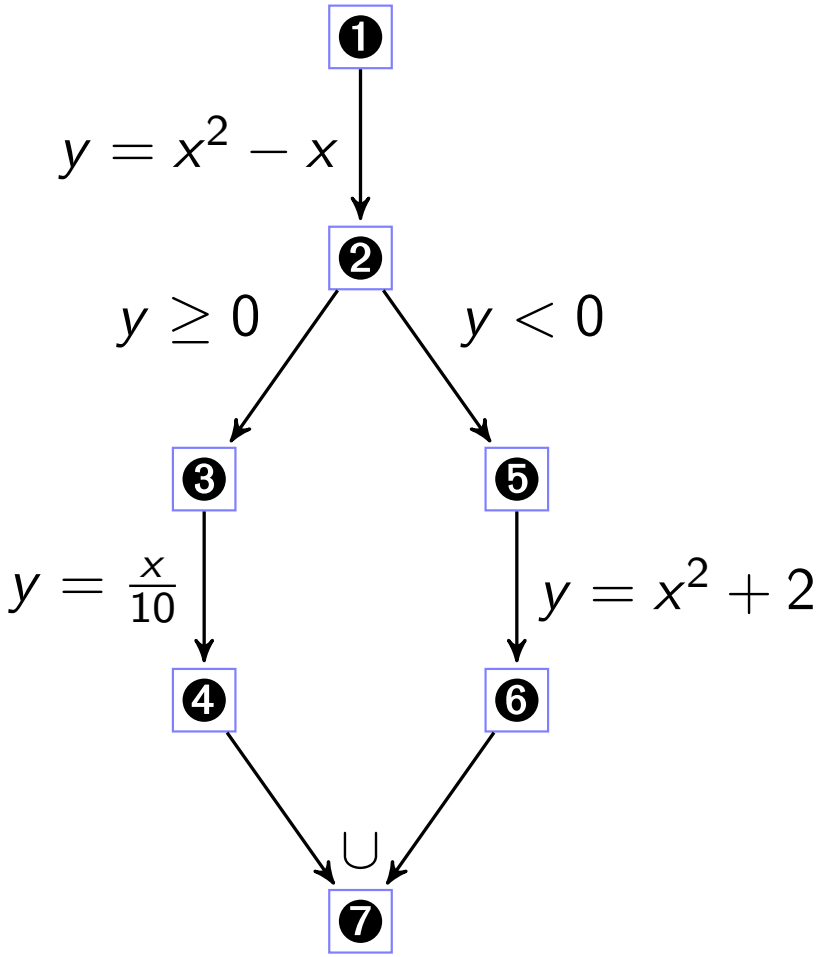
- run time errors: overflow, division by zero, square root of negatives, etc.
- robustness and stability of algorithms: linear and non linear recursive schemes, filters, etc.

- Program semantics formalized as a fixpoint of a monotonic operator in a complete partially ordered set (exemplified later),
- Fully automated,
- Industrial tools exists : Polyspace Verifier (MathWorks), AstrÃl'e (ENS/ABSINT), Fluctuat (CEA), aIT (ABSINT), F-Soft (Nec Labs) ...

Challenge

find the *suitable* abstract domain for the properties of interest.

Equations System (collecting semantic)



$$
\begin{cases}
\mathcal{X}_1 = [\![\mathcal{V} \to \mathbb{I}]\!]^\flat \\
\mathcal{X}_2 = [\![y \leftarrow x^2 - x]\!]^\flat(\mathcal{X}_1) \\
\mathcal{X}_3 = [\![y \geq 0]\!]^\flat(\mathcal{X}_2) \\
\mathcal{X}_4 = [\![y \leftarrow \frac{x}{10}]\!]^\flat(\mathcal{X}_3) \\
\mathcal{X}_5 = [\![y < 0]\!]^\flat(\mathcal{X}_2) \\
\mathcal{X}_6 = [\![y \leftarrow x^2 + 2]\!]^\flat(\mathcal{X}_5) \\
\mathcal{X}_7 = \mathcal{X}_6 \cup \mathcal{X}_4
\end{cases}
$$

Solving the equations system

- $D = (\wp(\mathcal{V} \to \mathbb{I}), \subseteq, \cup, \cap, \emptyset, (\mathcal{V} \to \mathbb{I}))$ is a **complete lattice**
- each operator $\mathcal{X} \mapsto \mathcal{F}(\mathcal{X})$ is monotonic

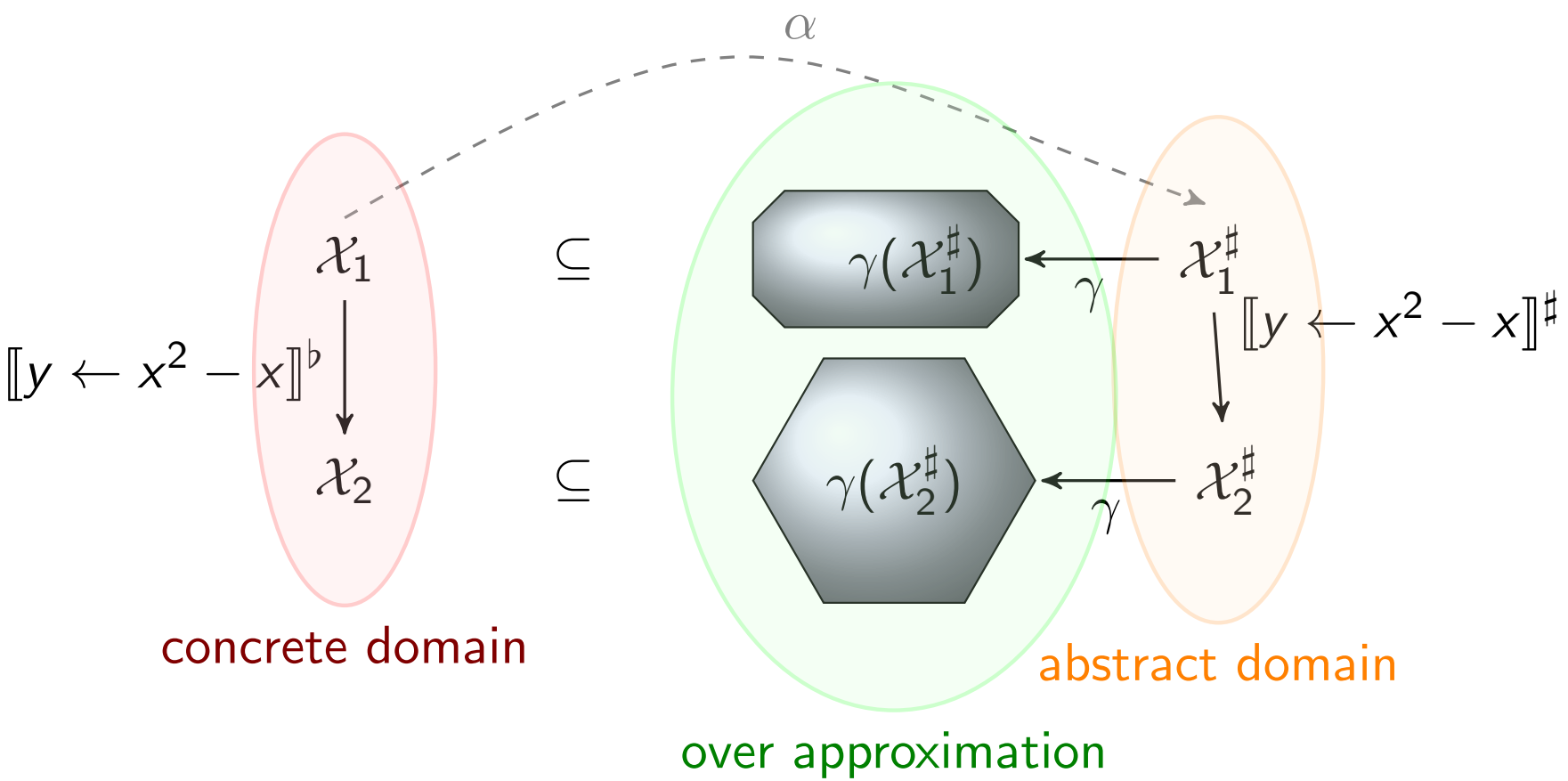
➙ **Tarski Theorem** ensures the existence of a least fixpoint for $\mathcal{F}$

➙ **Kleene Iteration Technique** reaches the least fixpoint

Issues

- $\wp(\mathcal{V} \to \mathbb{I})$ is non representable in finite memory,
- $[\![.]\!]^\flat$ are non computable,
- Iterations over the lattice may be transfinite.

Concretisation-Based Abstract Interpretation



Building an abstract domain

- lattice-like structure:
  - abstract objects
  - order relation (preorder over abstract objects)
  - monotonic concretisation function (γ)
- Transfer Functions
  - evaluation of arithmetic expressions ($[\![x^2 - x]\!]^\sharp$)
  - assignment ($\mathcal{X}_2 = [\![y \leftarrow x^2 - x]\!]^\sharp(\mathcal{X}_1)$)
  - upper bound (join) ($\mathcal{X}_7 = \mathcal{X}_6 \cup \mathcal{X}_4$)
  - over-approximation of lower bounds (meet) ($\mathcal{X}_3 = [\![y \geq 0]\!]^\sharp \mathcal{X}_2 =$ "$\mathcal{X}_3 = \mathcal{X}_2 \cap [\![y \geq 0]\!]^\sharp \top^\sharp$")
- Convergence acceleration (widening)