

Sweeping with Continuous Domains

G. Chabert and N. Beldiceanu

École des Mines de Nantes, LINA CNRS UMR 6241, FR-44300, France

`{gilles.chabert,nicolas.beldiceanu}@mines-nantes.fr`

CONTEXT AND MOTIVATION

FILTERING WITH SWEEP

A GENERIC INFLATER FOR ARITHMETICAL CONSTRAINTS

CONCLUSION

Sweep Algorithms in Computational Geometry

Standard technique in the design of **efficient** algorithms, described in:

- Computational geometry, an introduction
[Preparata & Shamos, 1985]
- Computational Geometry, Algorithms and Applications
[Berg, Kreveld, Overmars & Schwarzkopf, 1997]
- Géométrie algorithmique
[Boissonnat & Yvinec, 1995]

Sweep Algorithms

Within the **Geometry Literature Database** (*more than 100 references*)

- Voronoi diagram
- map overlay
- nearest objects
- triangulations
- hidden surface removals
- rectangles intersection
- shortest path

Within **Constraint Programming**

- conjunction of constraints
 - **cardinality** operator
 - multi resource **cumulatives**
 - the **spread** constraint
 - **non-overlapping** constraints
(*boxes, polygons, symmetry*)
 - Formulas (**geost rules**)
(*quantifier-free Presburger arithmetic*)
-

Observations and Motivations

Currently, within constraint programming, **sweep** is mostly used:

- for handling **geometrical** constraints
(and more specifically non-overlapping between boxes)
- with **discrete** variables
(the coordinates of the boxes)
- it handles **disjunction**
(there is a least one dimension where projections do not overlap)
- it takes advantage of the **clique** of constraints
(and the way constraints share variables)

Observation

In this context, infeasible points (*and feasible points*) are usually grouped together (we have **clouds of infeasible points**, *different from making intersection between curves*).

Motivation for this work

Provide **a building block** for handling in a generic way geometrical constraints between, e.g., boxes, circles, ellipses, spheres, cylinders with continuous variables.

CONTEXT AND MOTIVATION

 **FILTERING WITH SWEEP**

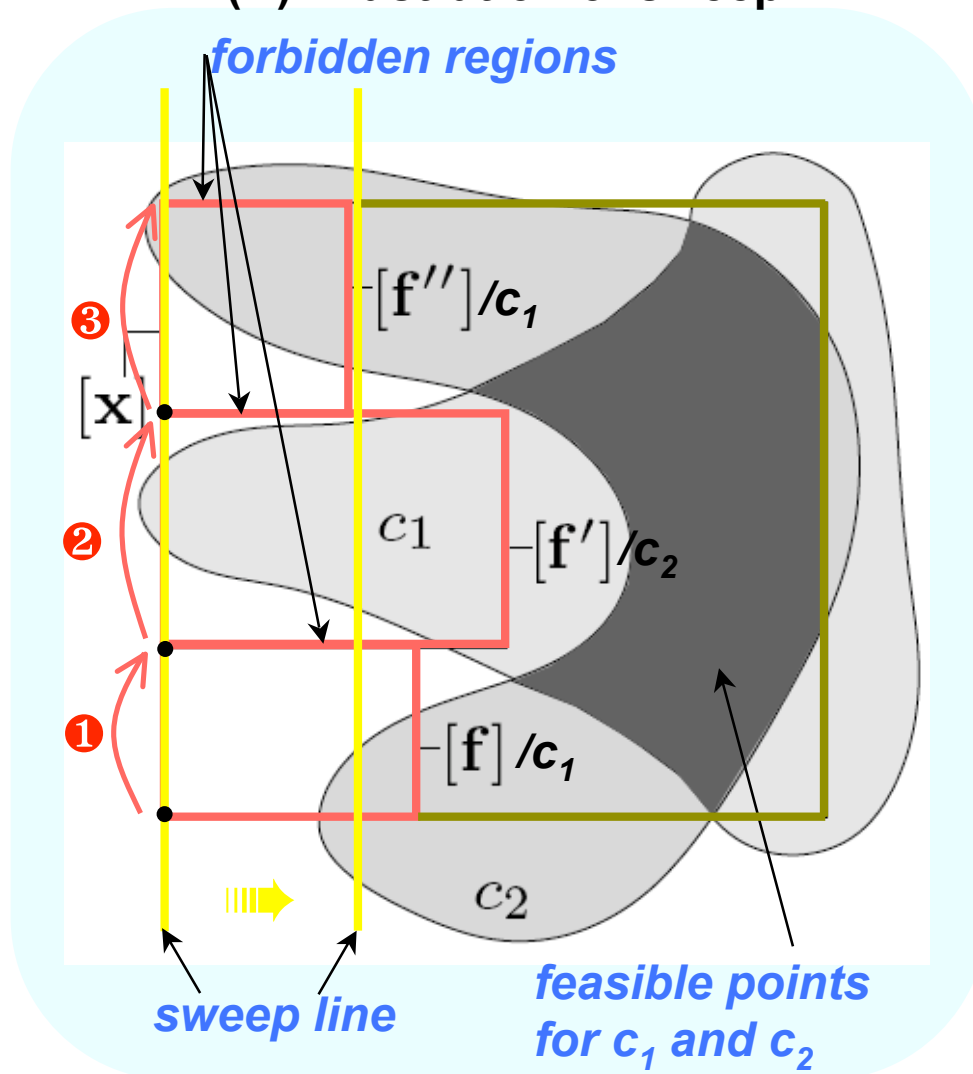
A GENERIC INFLATER FOR ARITHMETICAL CONSTRAINTS

CONCLUSION

Basic Idea of Sweep

Aggregating forbidden regions (*box containing only infeasible points for a ctr.*) :

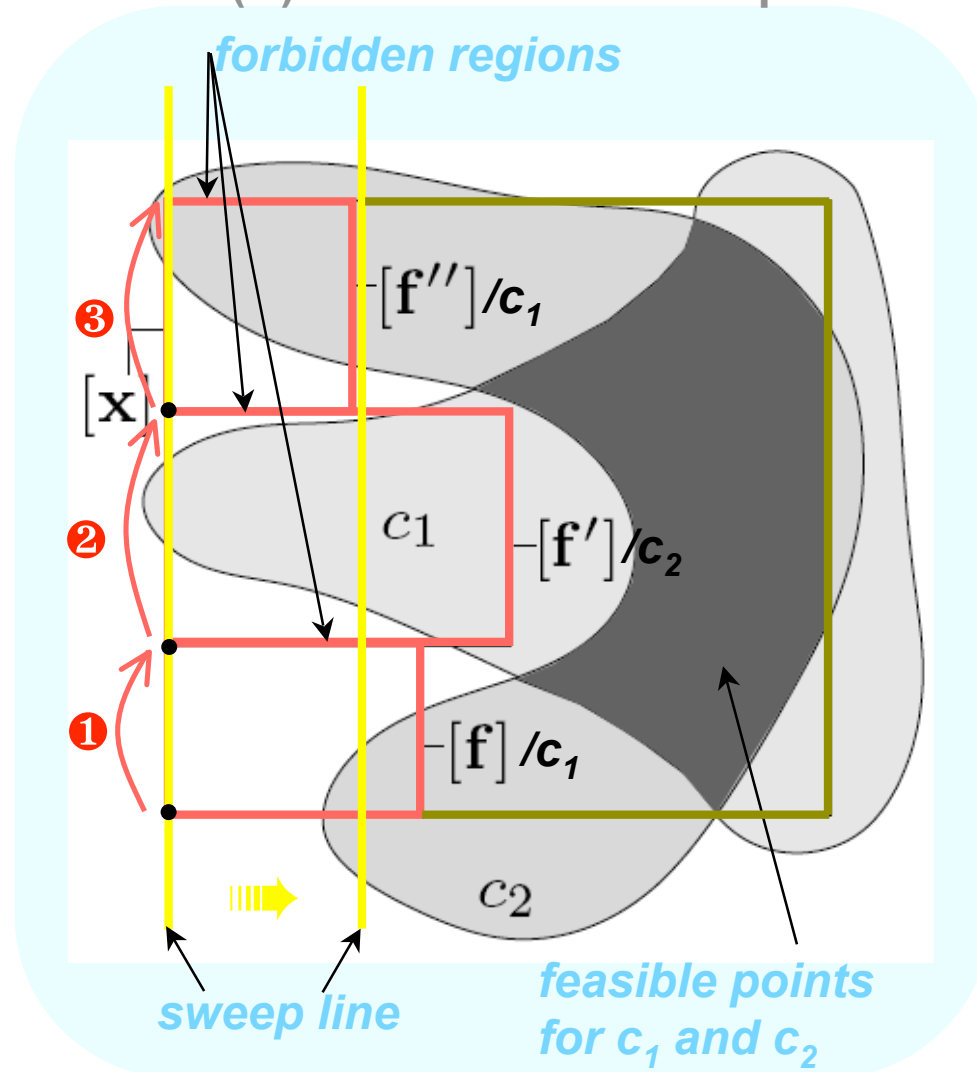
(A) Illustration of sweep



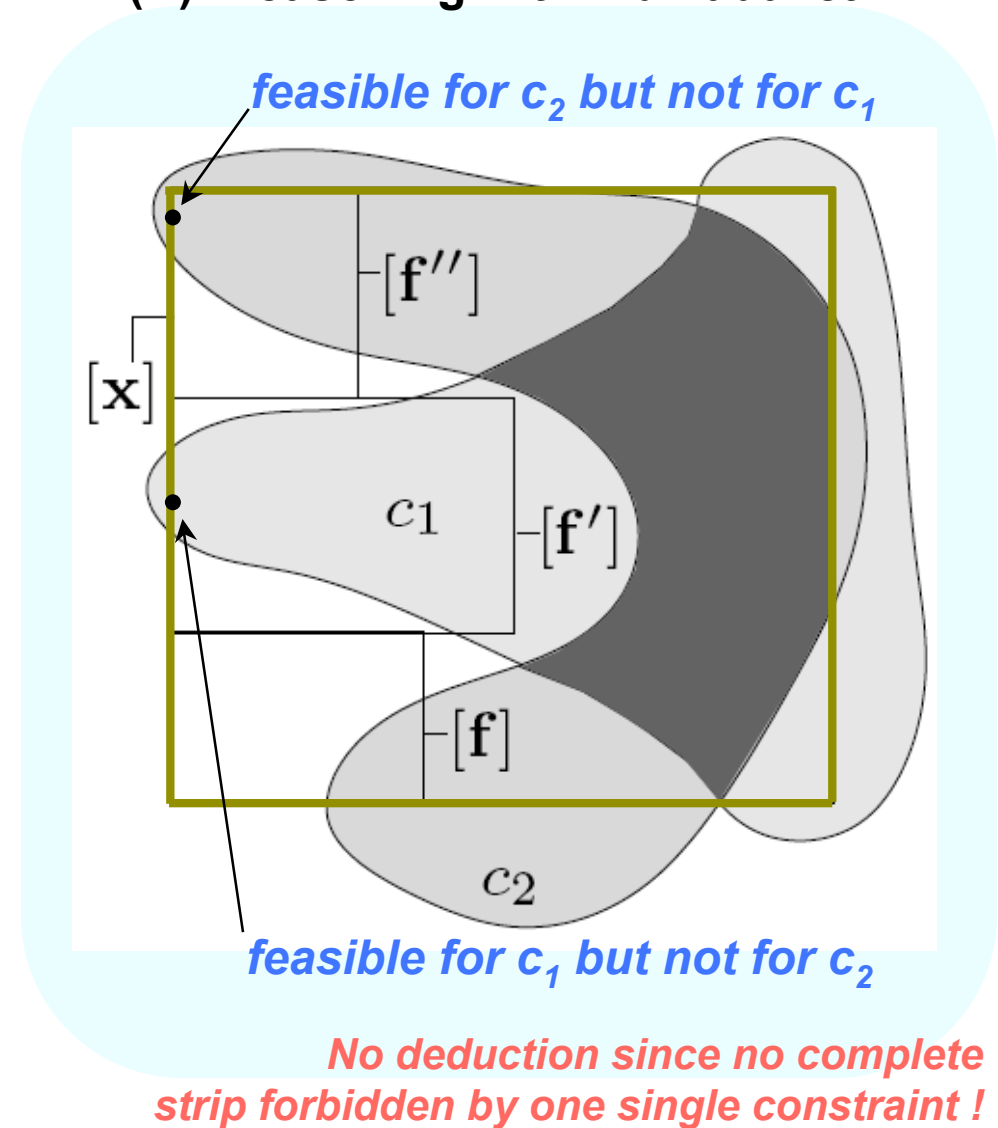
Basic Idea of Sweep

Aggregating forbidden regions (box containing only infeasible points for a ctr.) :

(A) Illustration of sweep



(B) Reasoning with individual ctr.



Basic service required by sweep: a generic **inflater**

Given:


- (1) a **constraint** $C(V_1, V_2, \dots, V_n)$,
- (2) an **infeasible assignment** (v_1, v_2, \dots, v_n) for C ,
- (3) a **direction** dir (+ or -),

computes a box B that

- (4) B contains (v_1, v_2, \dots, v_n) ,
- (5) B contains **only** infeasible points,
- (6) if dir is + then (v_1, v_2, \dots, v_n) is the **lower corner** of B ,
- (7) if dir is - then (v_1, v_2, \dots, v_n) is the **upper corner** of B .

the constraint C can be:

- an adhoc constraint,
- a logical expression (*disjunction of linear inequalities*),
- a formula (*geost rule*),

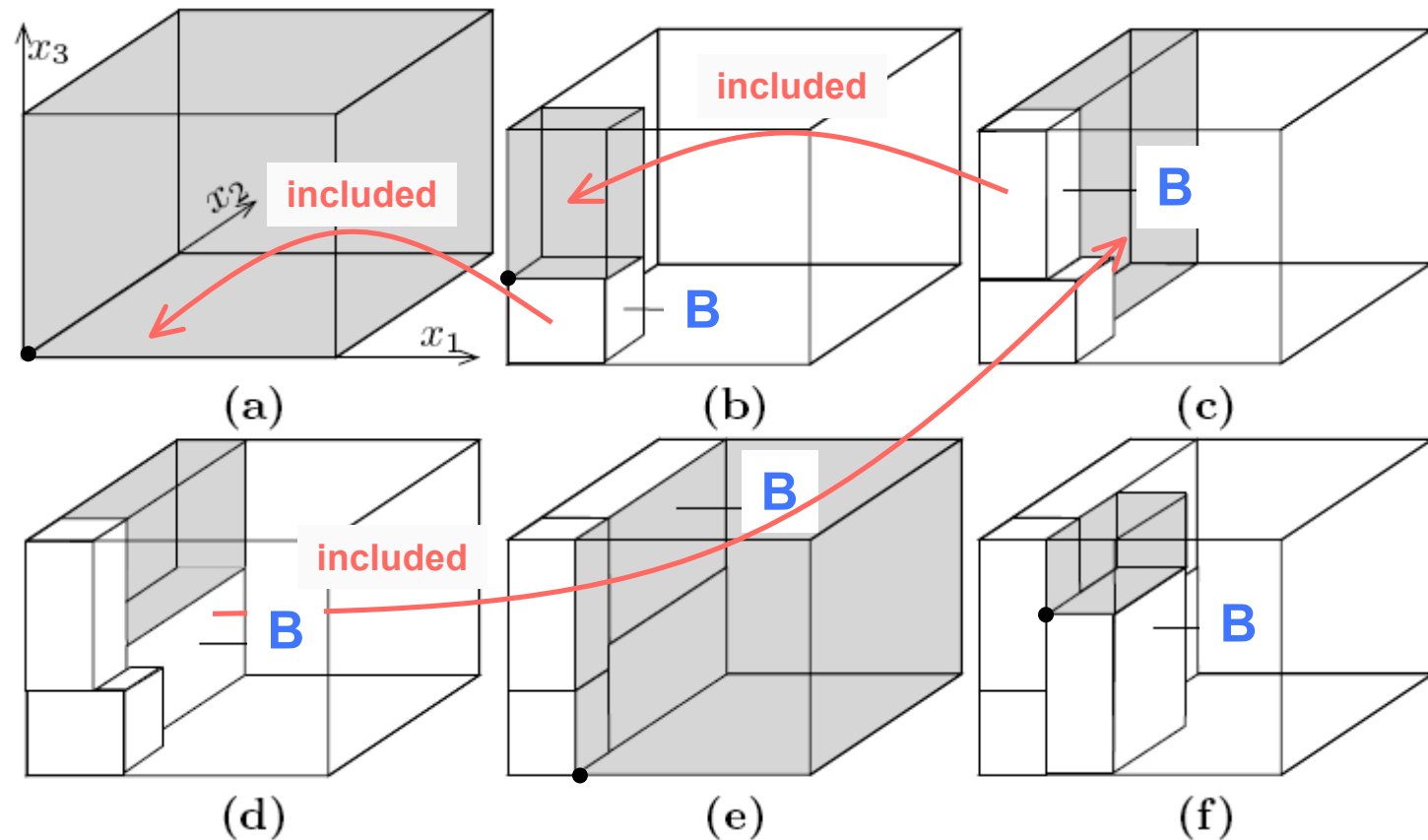
this paper  • a formula (*numerical constraints with continuous variables*)

Working area of B

IDEA: In some dimensions, **B** is **restricted** by the previous forbidden boxes

Additional requirement :

(8) **B** should be contained in the working area (*in grey on the example*)

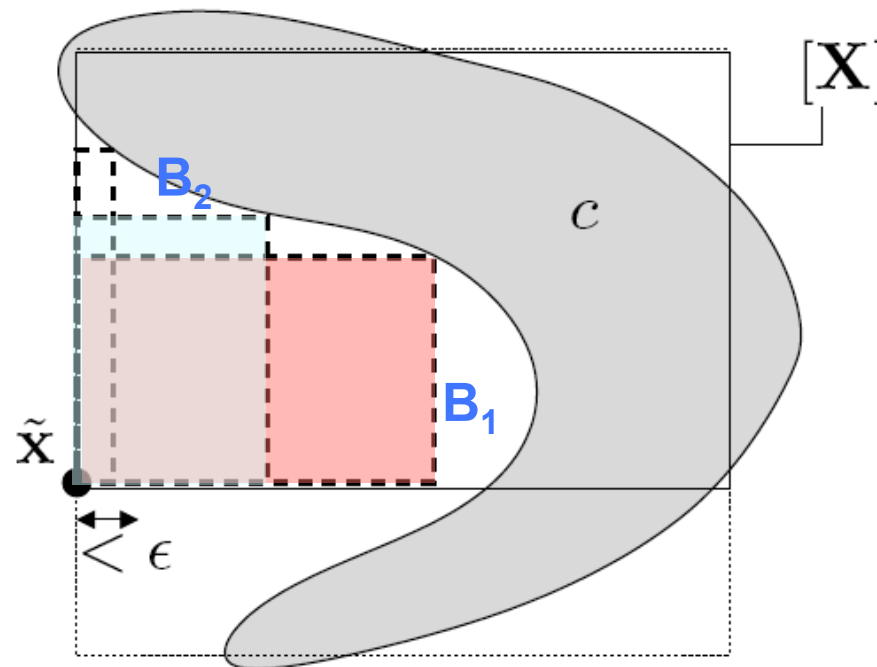


Importance of the Working Area in a Continuous Setting

INTUITION: Reduce the risk for generating degenerated forbidden boxes
(i.e., boxes that are wide in one dimension but very narrow
in an other)

What an inflater does:

Take a forbidden point \tilde{x} w.r.t. c
and build a forbidden box B , as
large as possible, around \tilde{x}
and **inside** the working area.



non-unicity and quality of inflation

CONTEXT AND MOTIVATION

FILTERING WITH SWEEP

➔ A GENERIC INFLATER FOR ARITHMETICAL CONSTRAINTS

CONCLUSION

Inflator Algorithm: intuition

Given,

- the **syntactical tree** of the mathematical expression of the constraint (*where intermediate variables are created at each level of the tree*),
- an **infeasible point** and a **working area**,

A two phases algorithm:

1) A **forward phase** propagates up to the root:

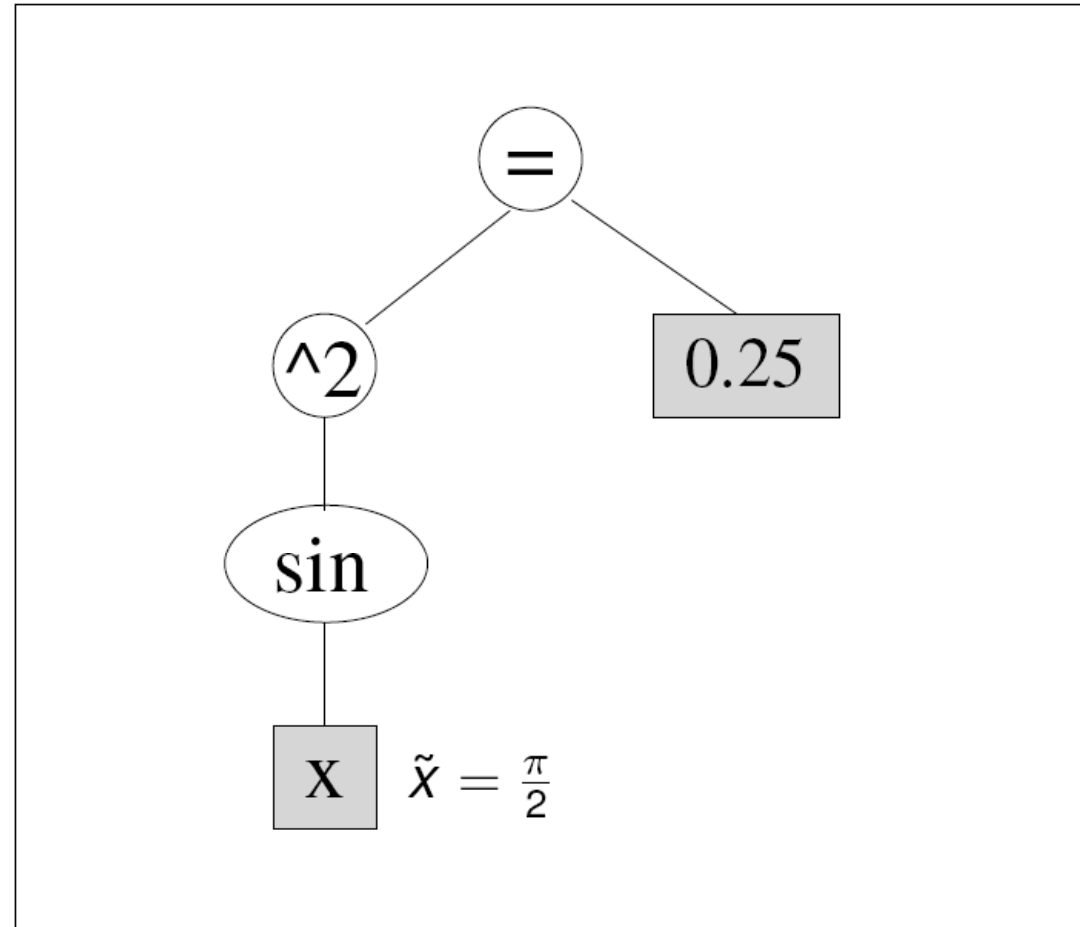
- the coordinates of the infeasible point,
- the intervals corresponding to the working area.

2) A **backward phase** propagates down to the leaves:

- the fact that the constraint is not satisfied (*since want to compute a box containing only infeasible points*)

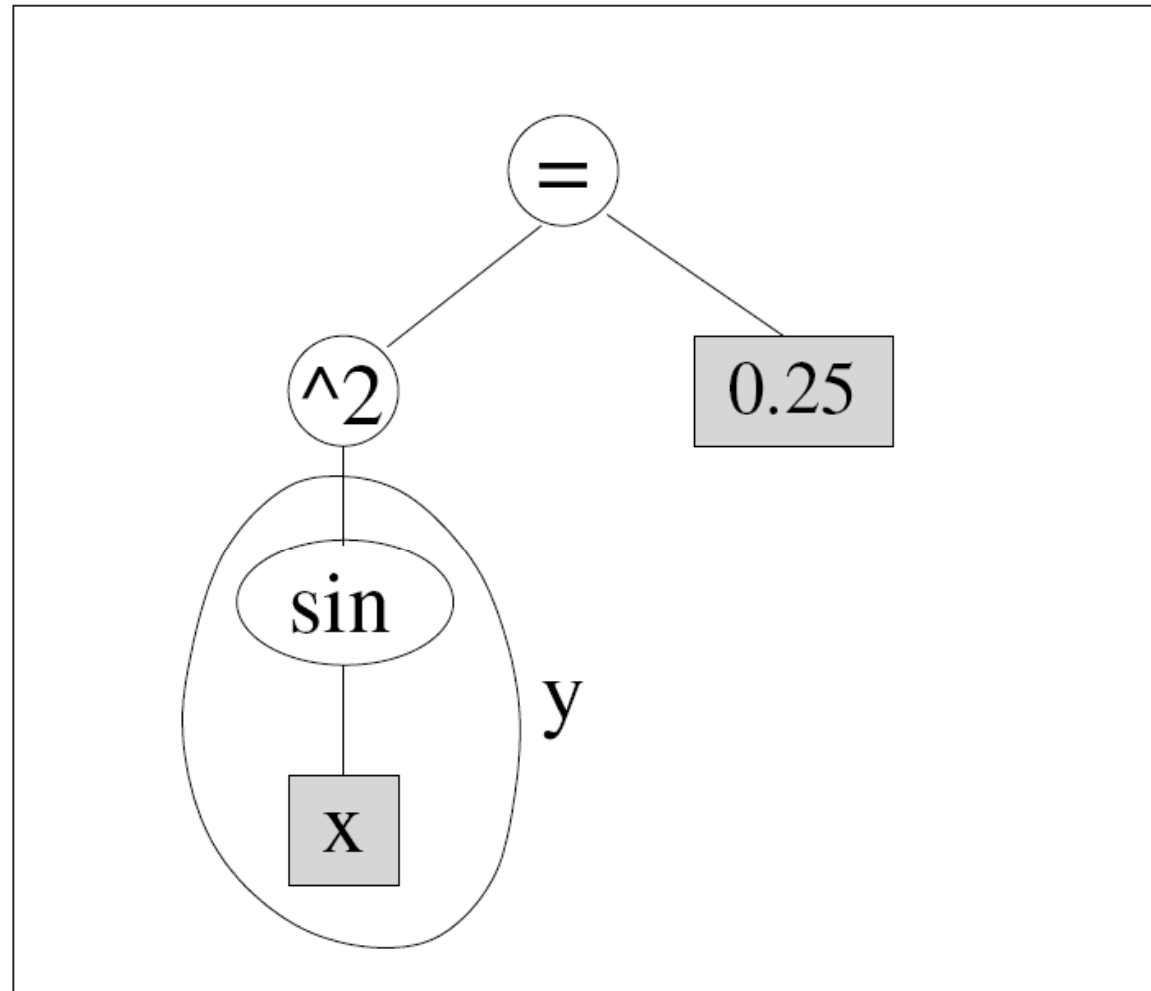
The information computed during the forward phase guides selecting intervals that contain the coordinates of the infeasible point.

Main Algorithm (example)



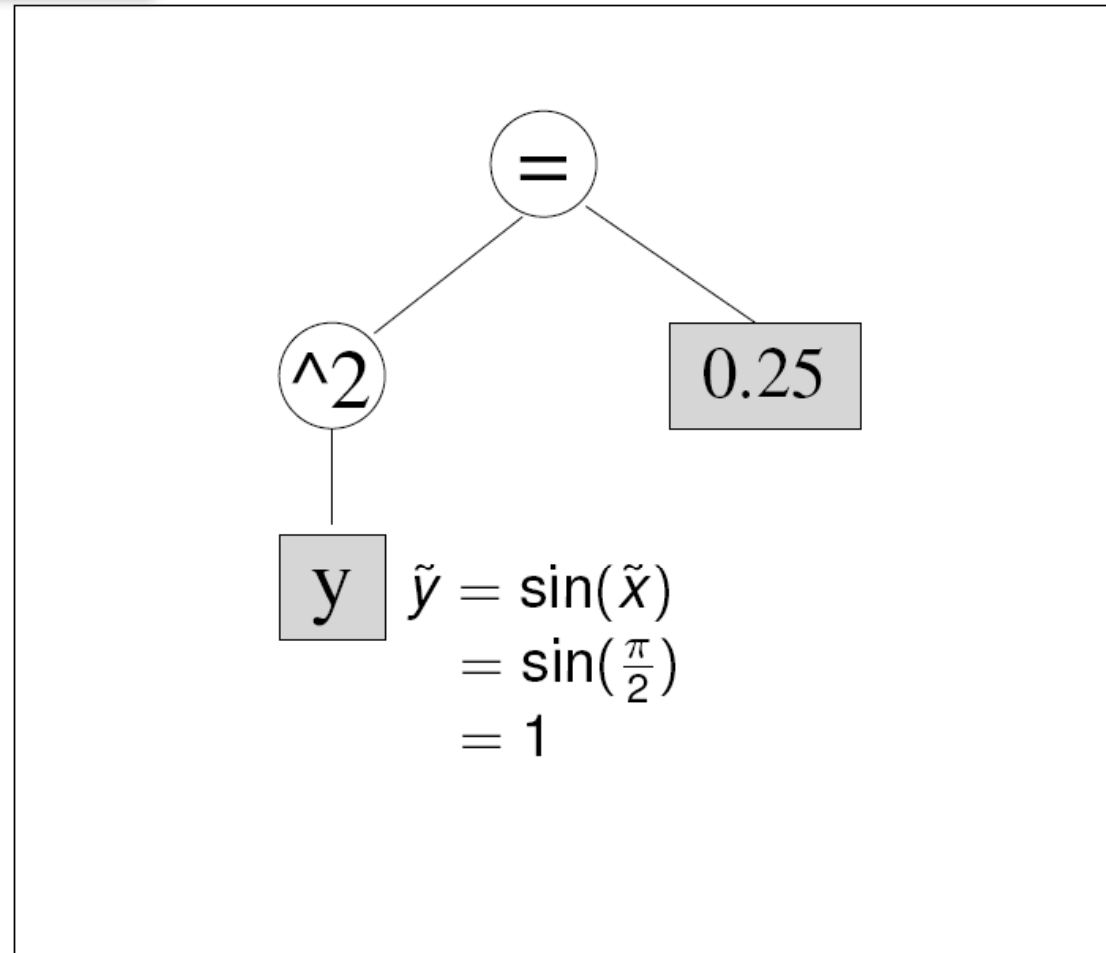
Goal: compute a forbidden interval $[x]$ around $\tilde{x} = \frac{\pi}{2}$ w.r.t. the constraint $\sin(x)^2 = 0.25$

Main Algorithm (example)



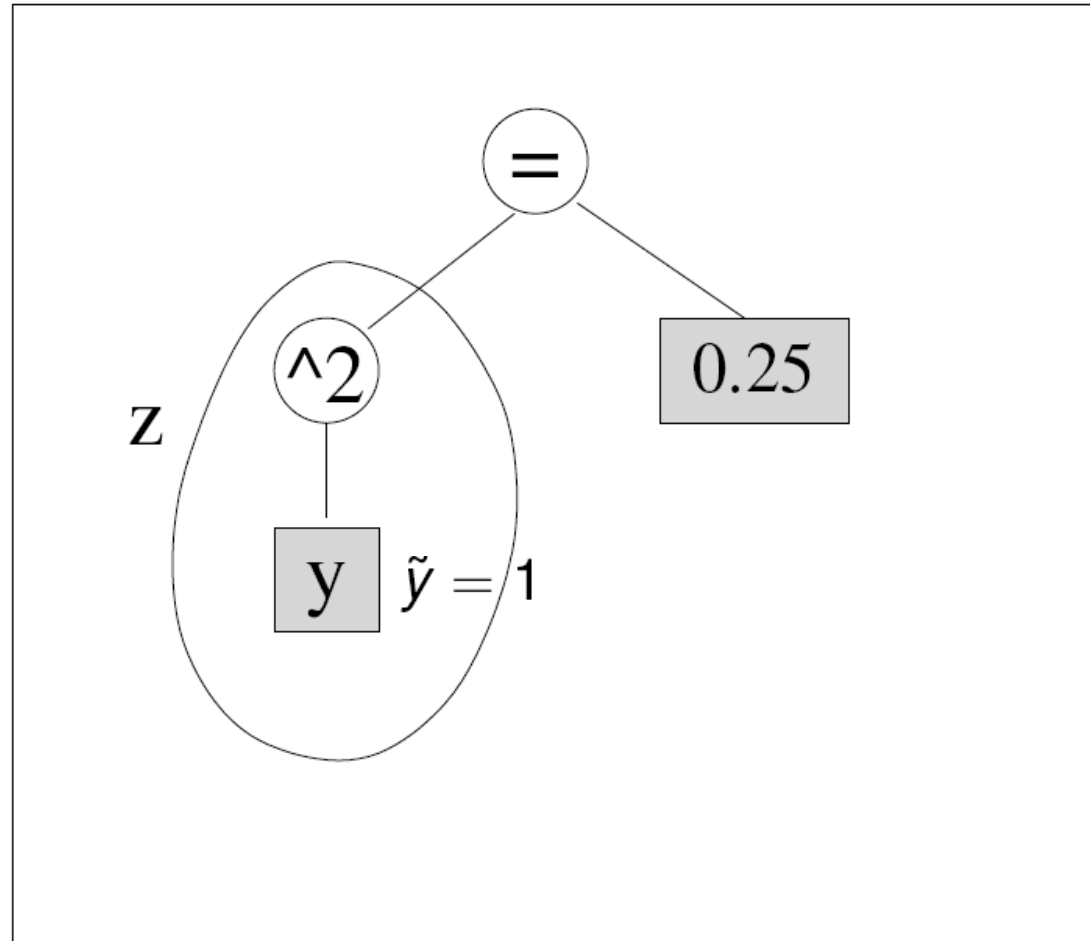
Introduce an intermediate variable $y = \sin(x)$

Main Algorithm (example)



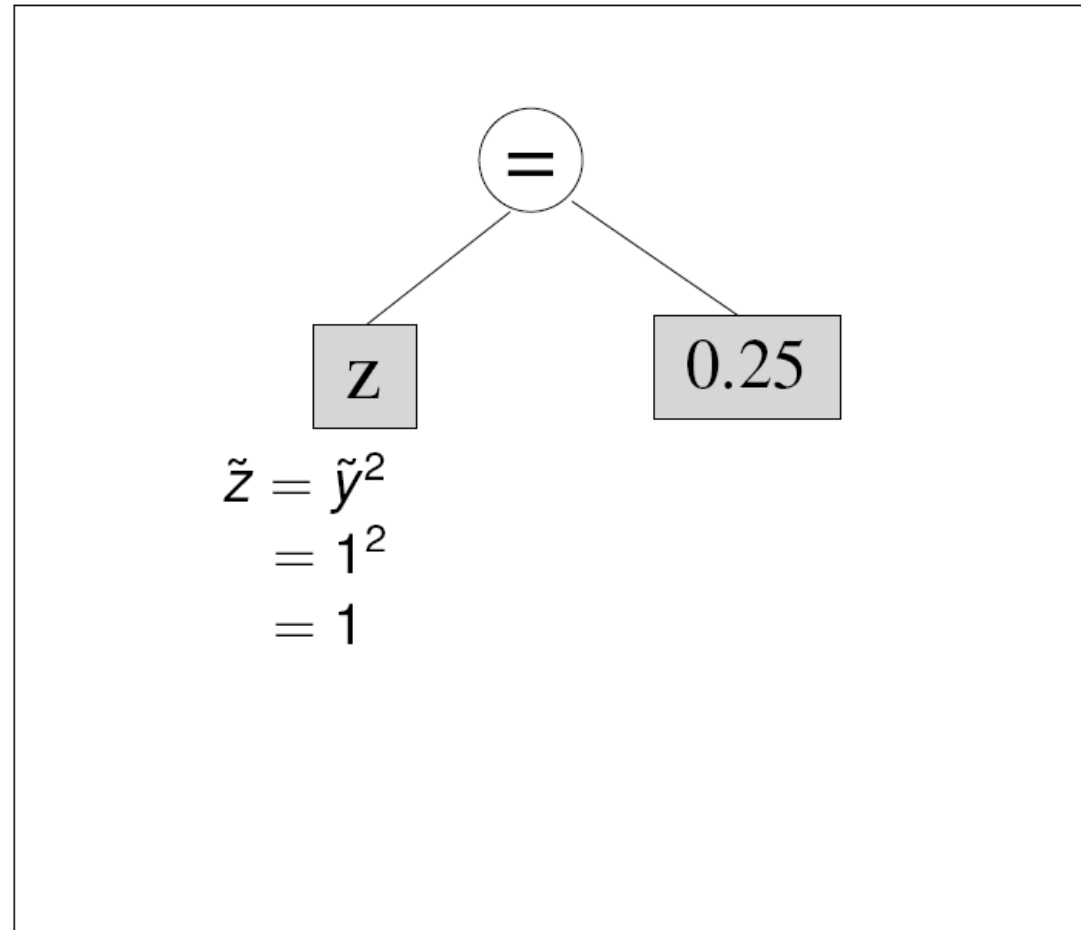
and calculate \tilde{y} , the image of \tilde{x} . The subtree $\sin(x)$ is replaced by a new node y .

Main Algorithm (example)



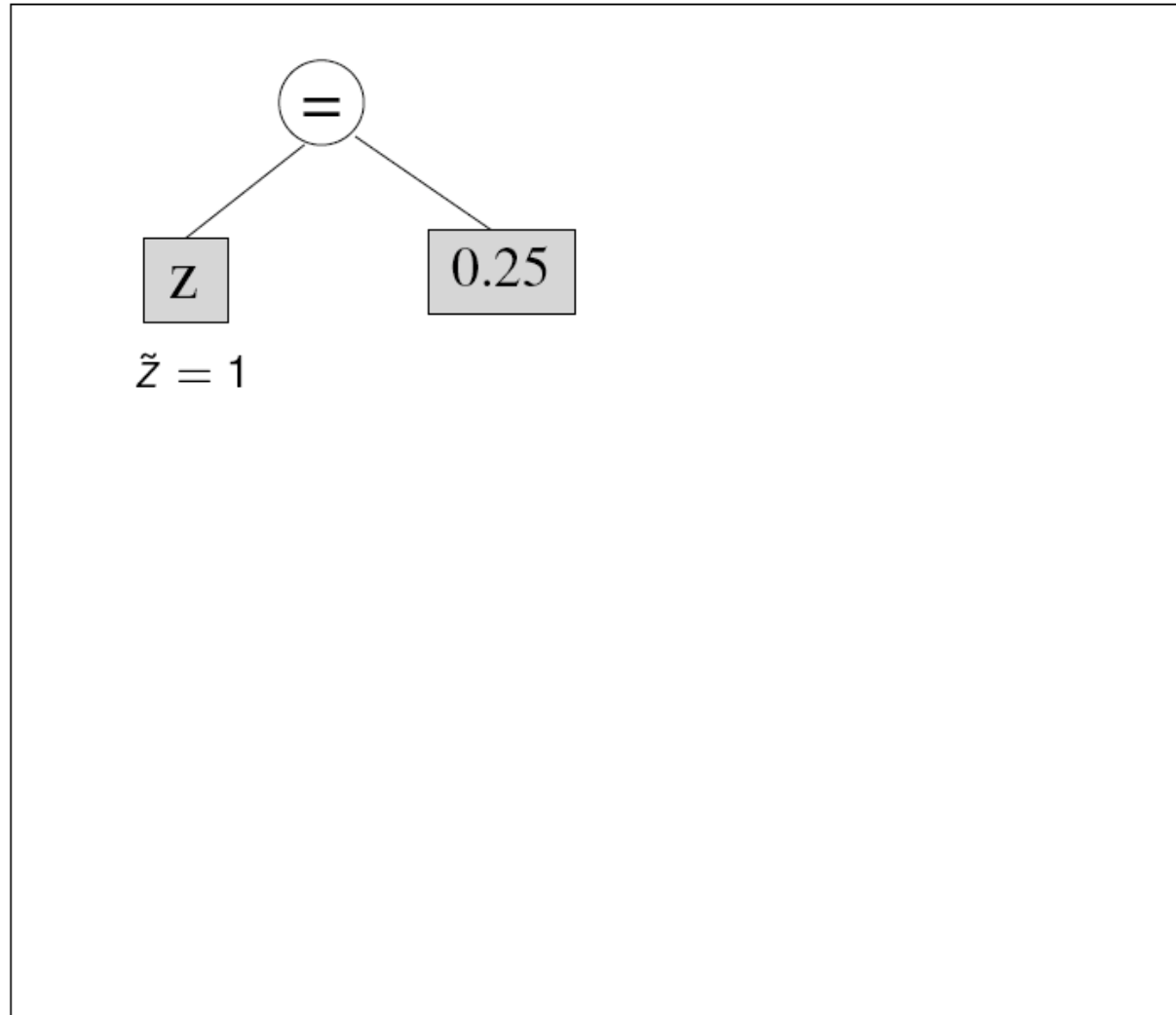
Repeat this substitution procedure up to the root of the tree.

Main Algorithm (example)



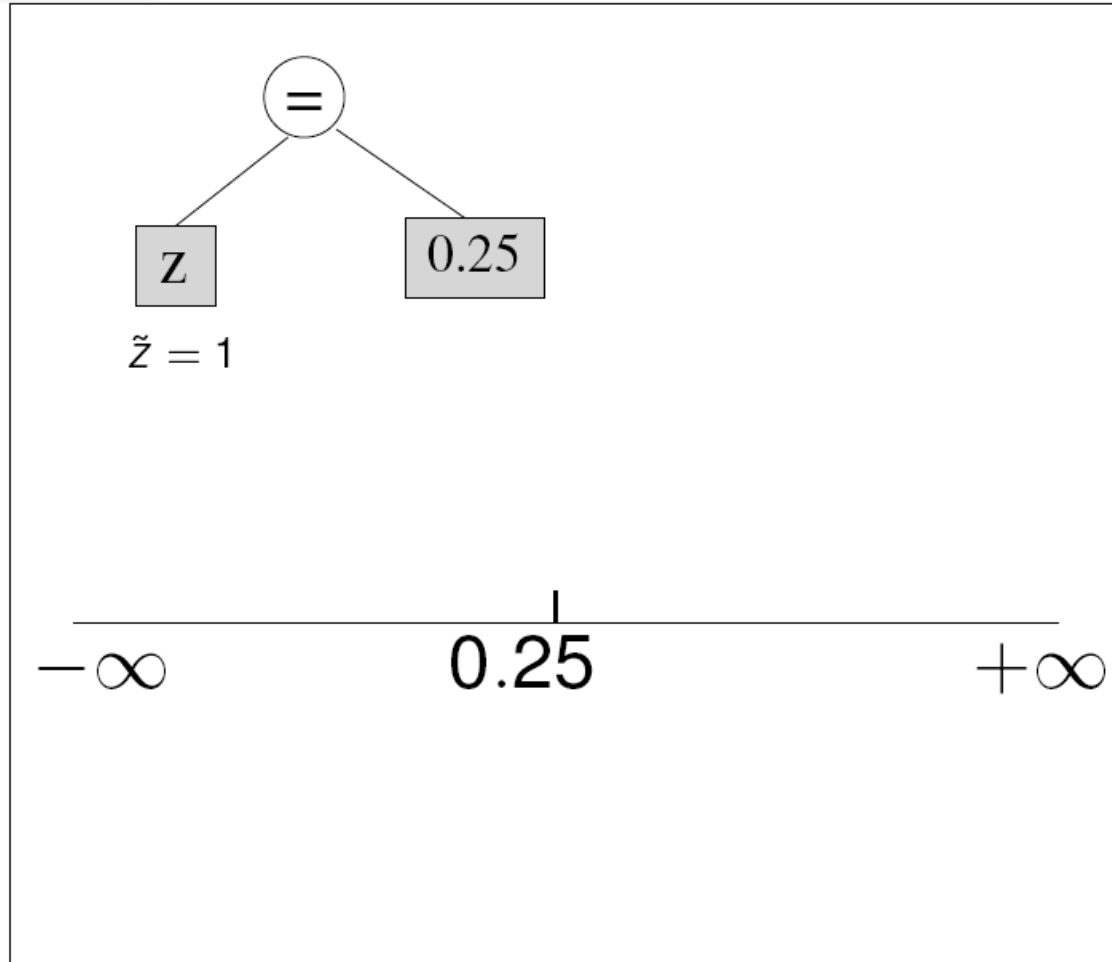
Repeat this substitution procedure up to the root of the tree.

Main Algorithm (example)



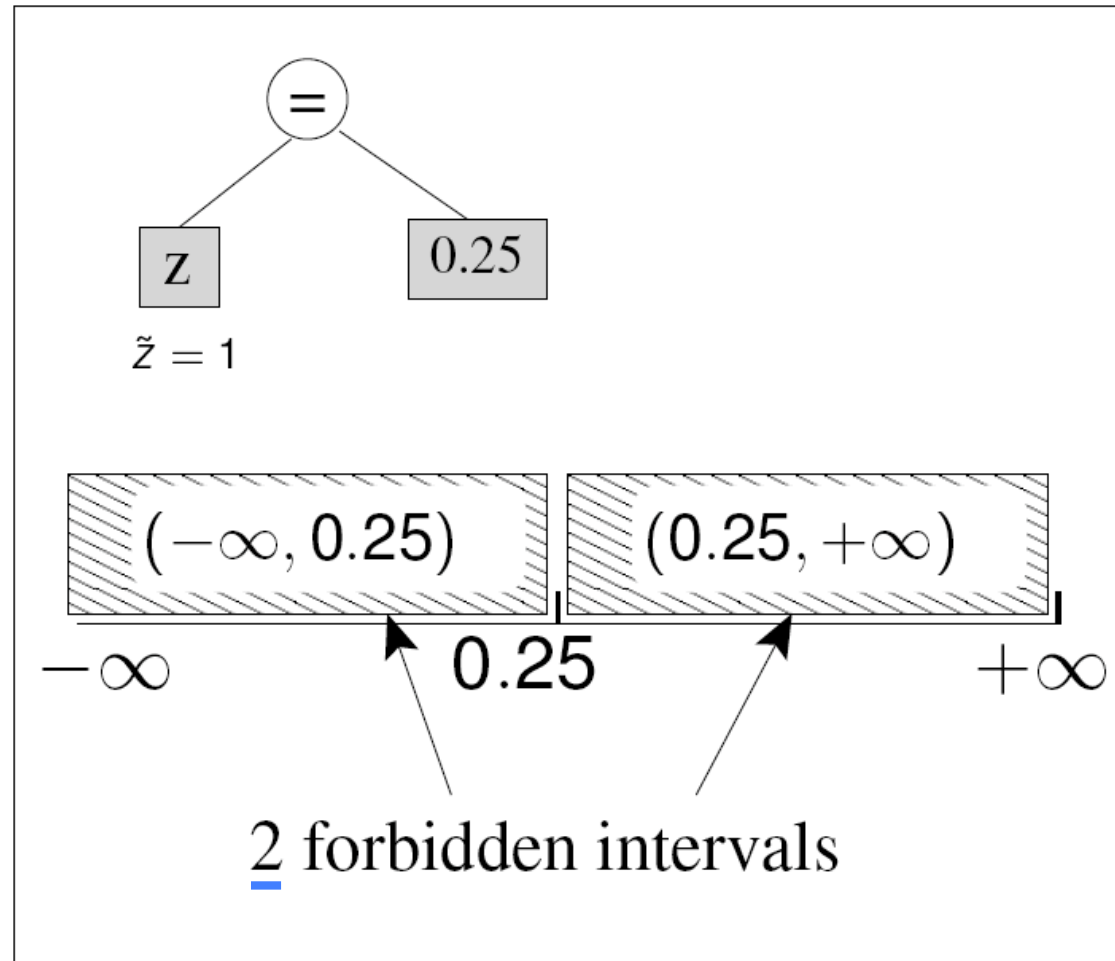
Backward step...

Main Algorithm (example)

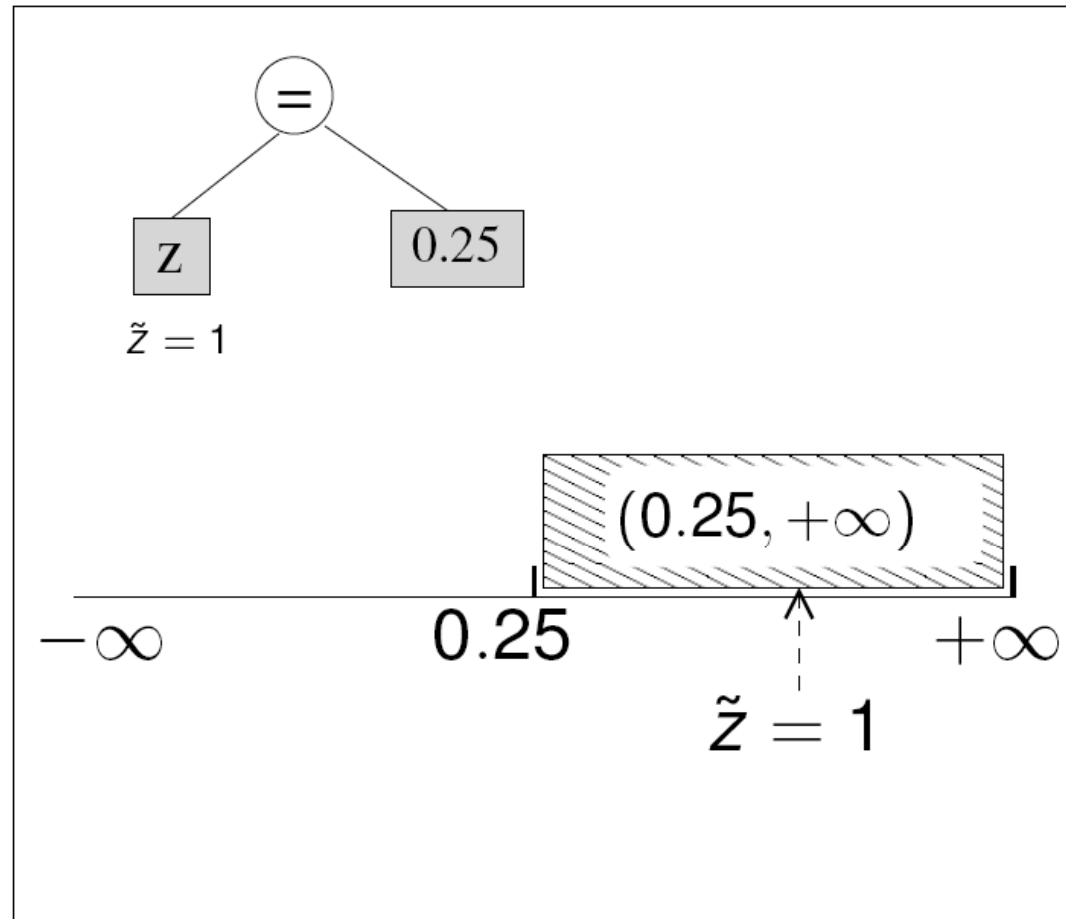


Find a forbidden interval w.r.t the elementary constraint
 $z = Cte$

Main Algorithm (example)

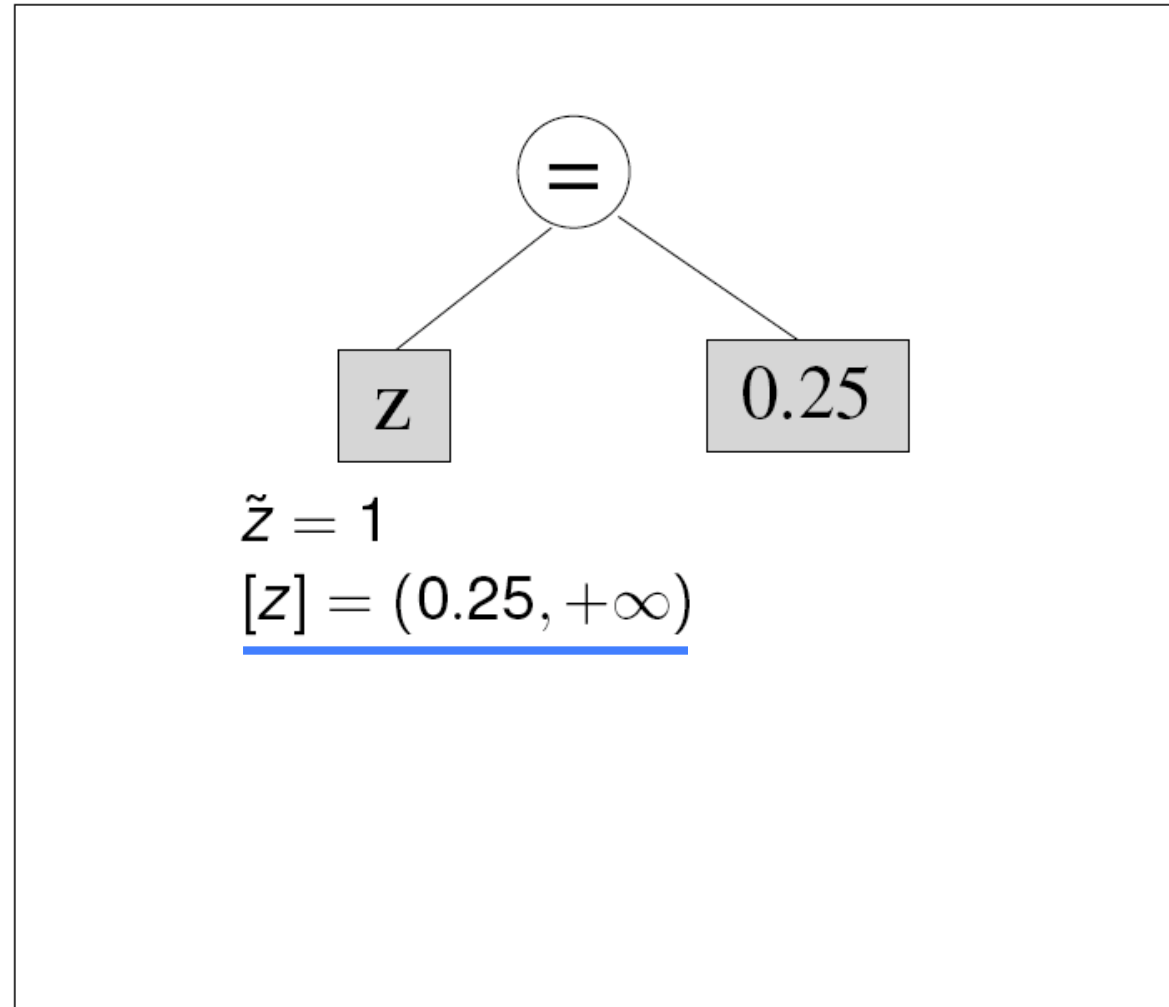


Main Algorithm (example)

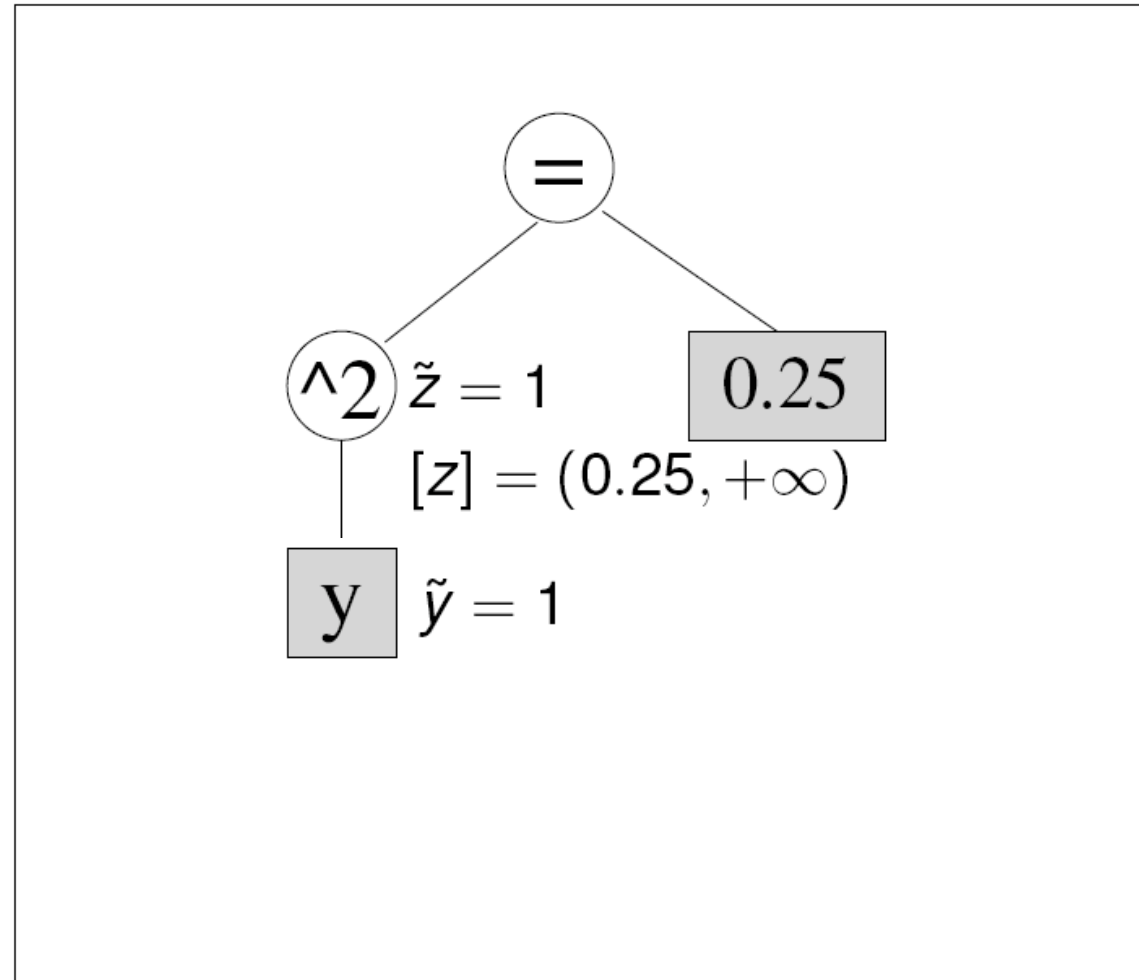


Select the unique interval that can contain \tilde{z} , i.e., the image of \tilde{x}

Main Algorithm (example)

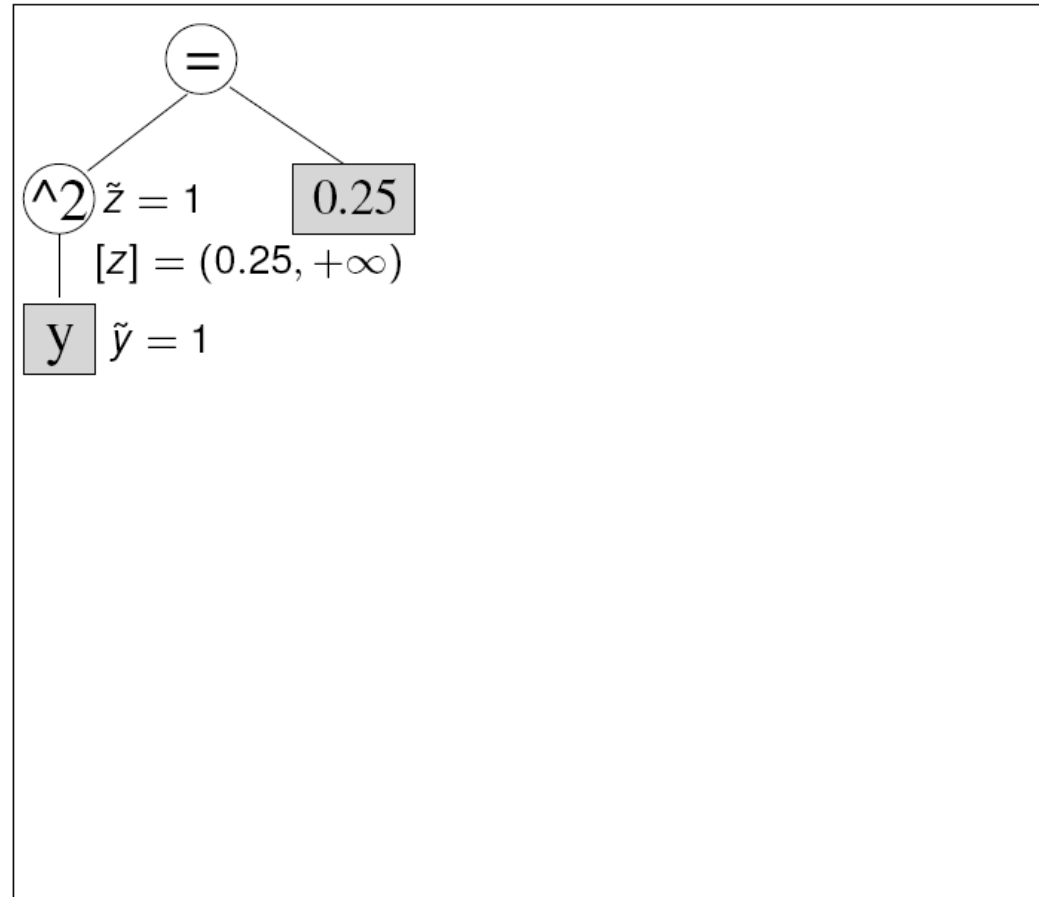


Main Algorithm (example)



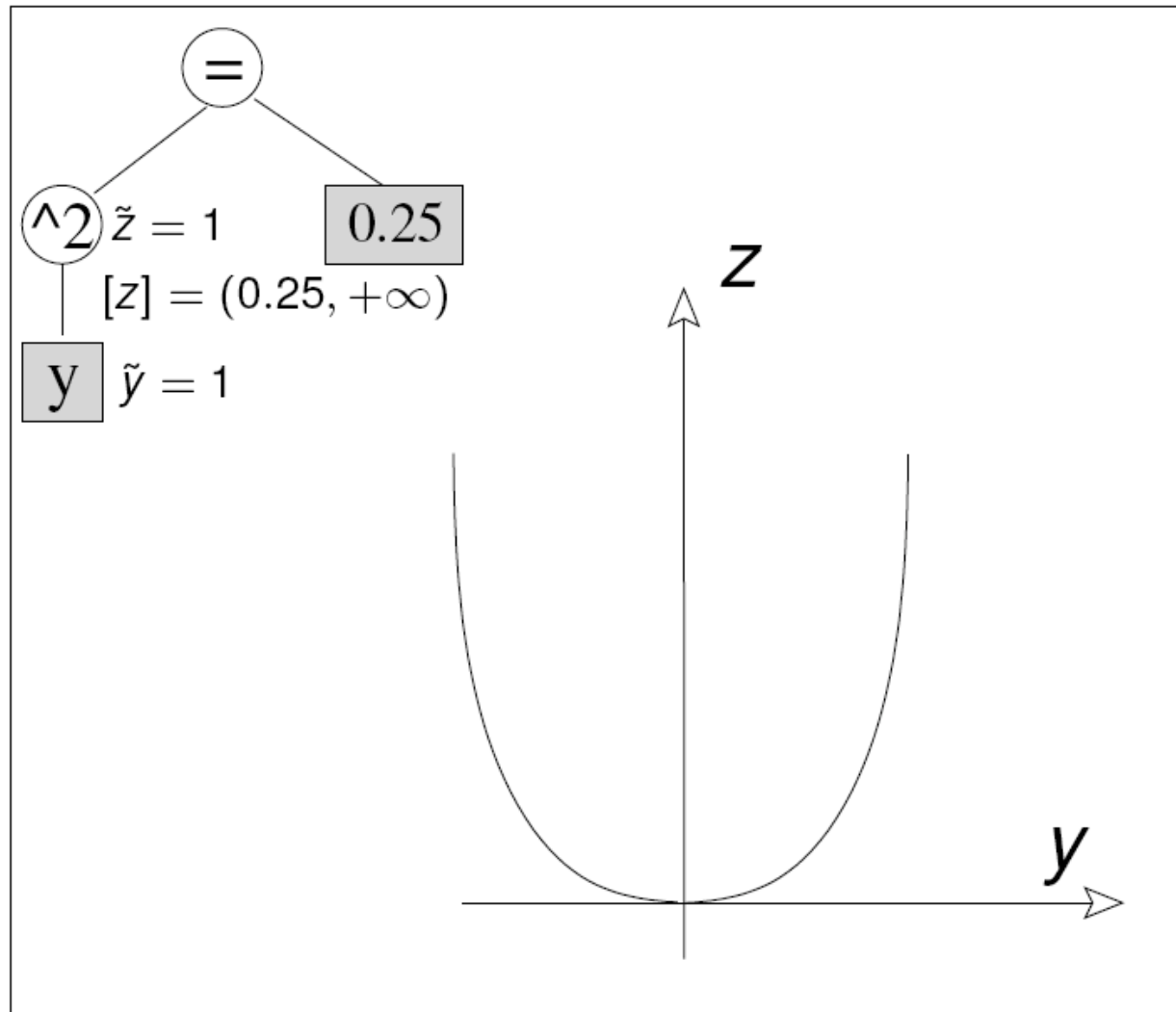
“Unfold” the node z

Main Algorithm (example)

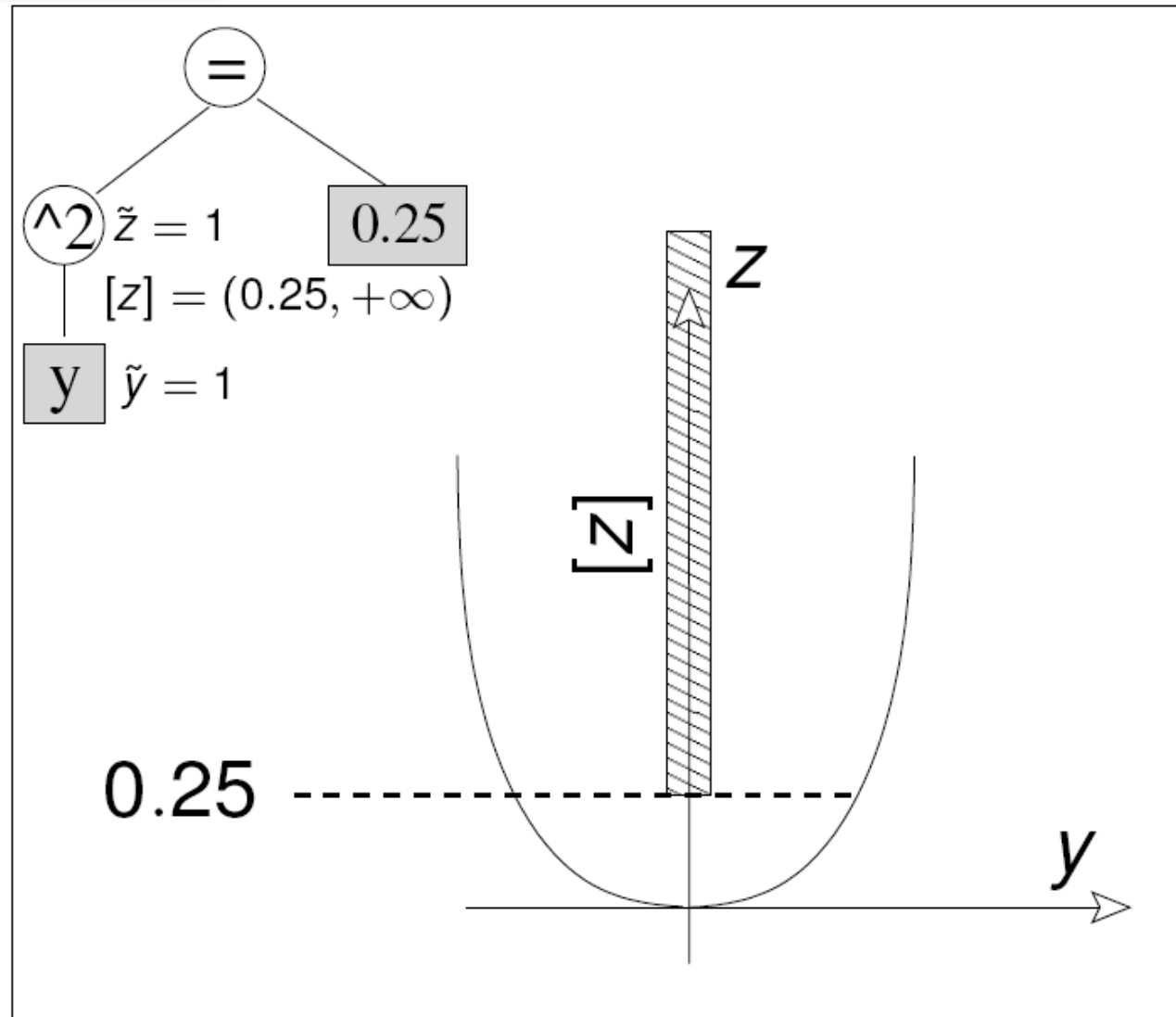


We must find now a forbidden interval for y with the knowledge that $y^2 \in (0.25, +\infty)$ is forbidden

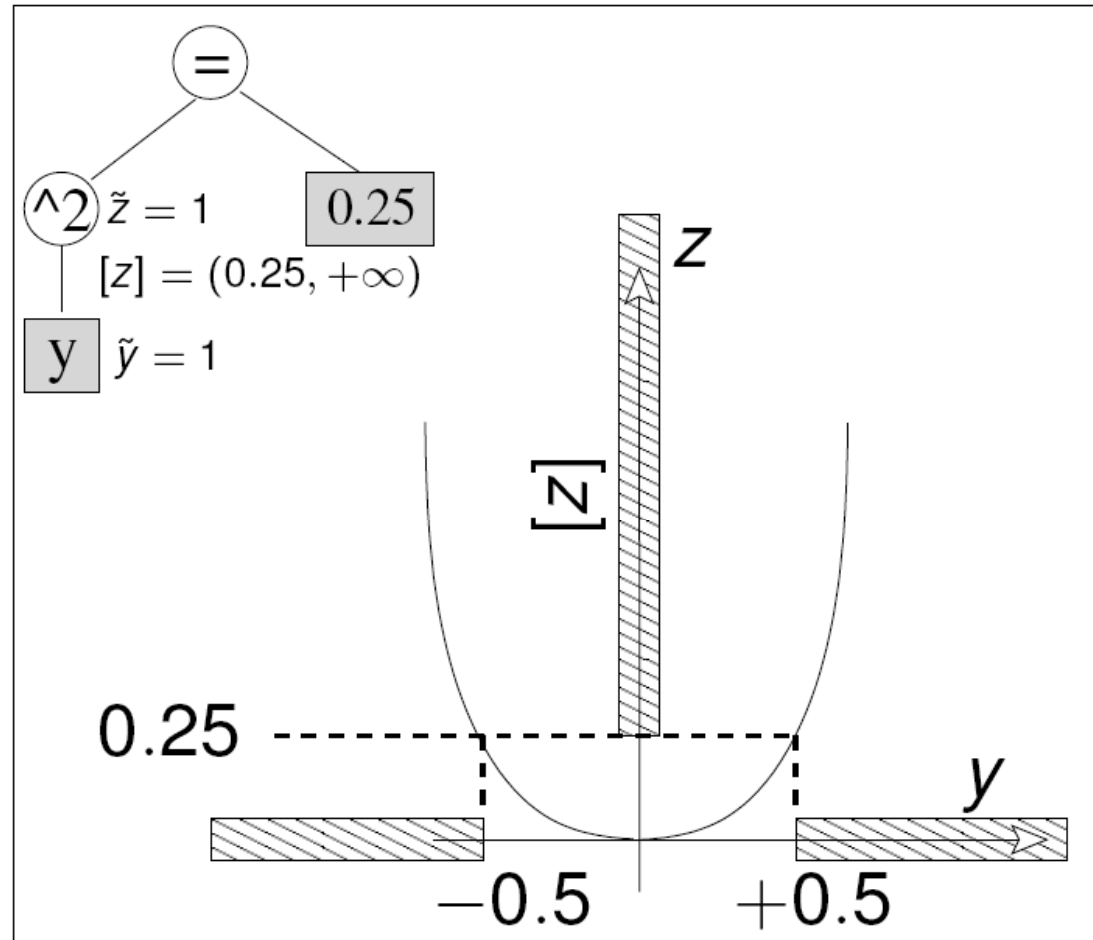
Main Algorithm (example)



Main Algorithm (example)

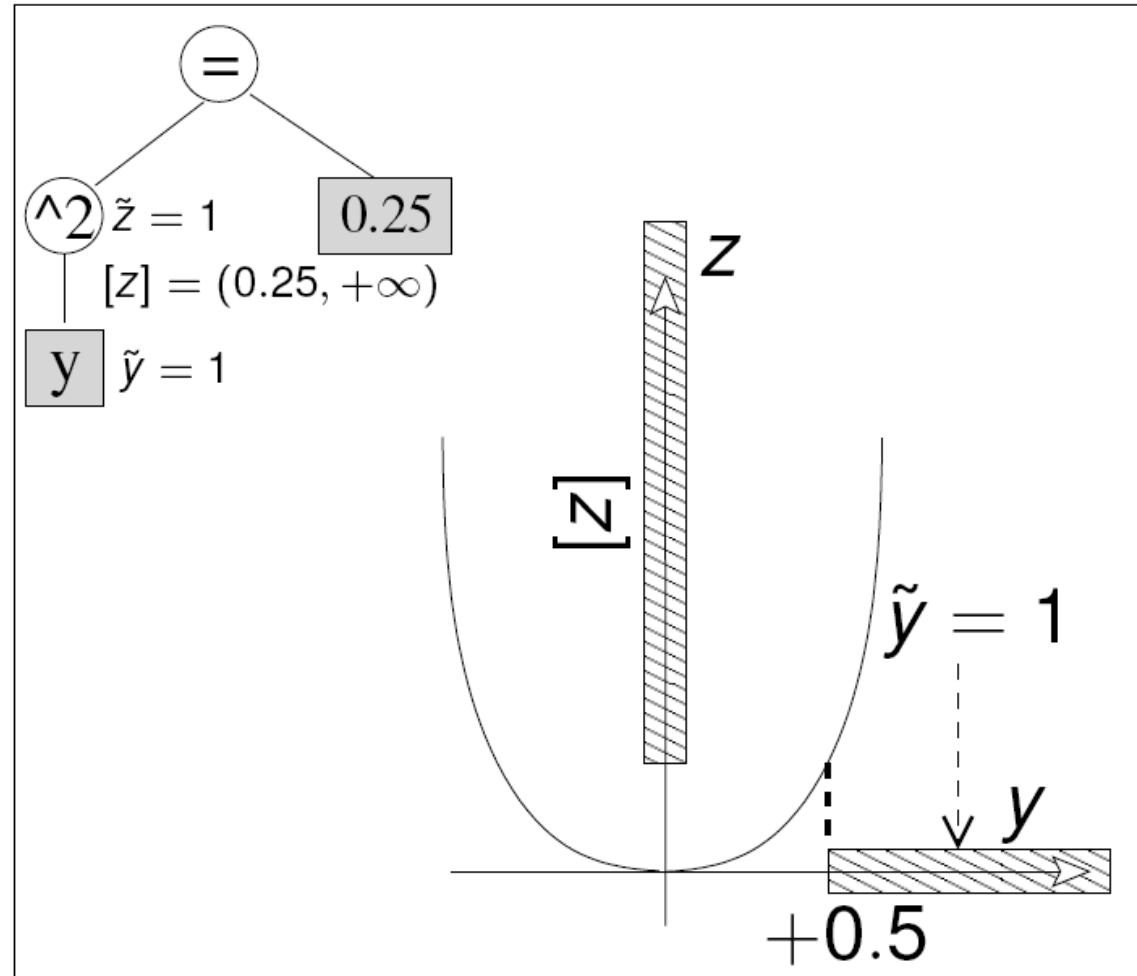


Main Algorithm (example)



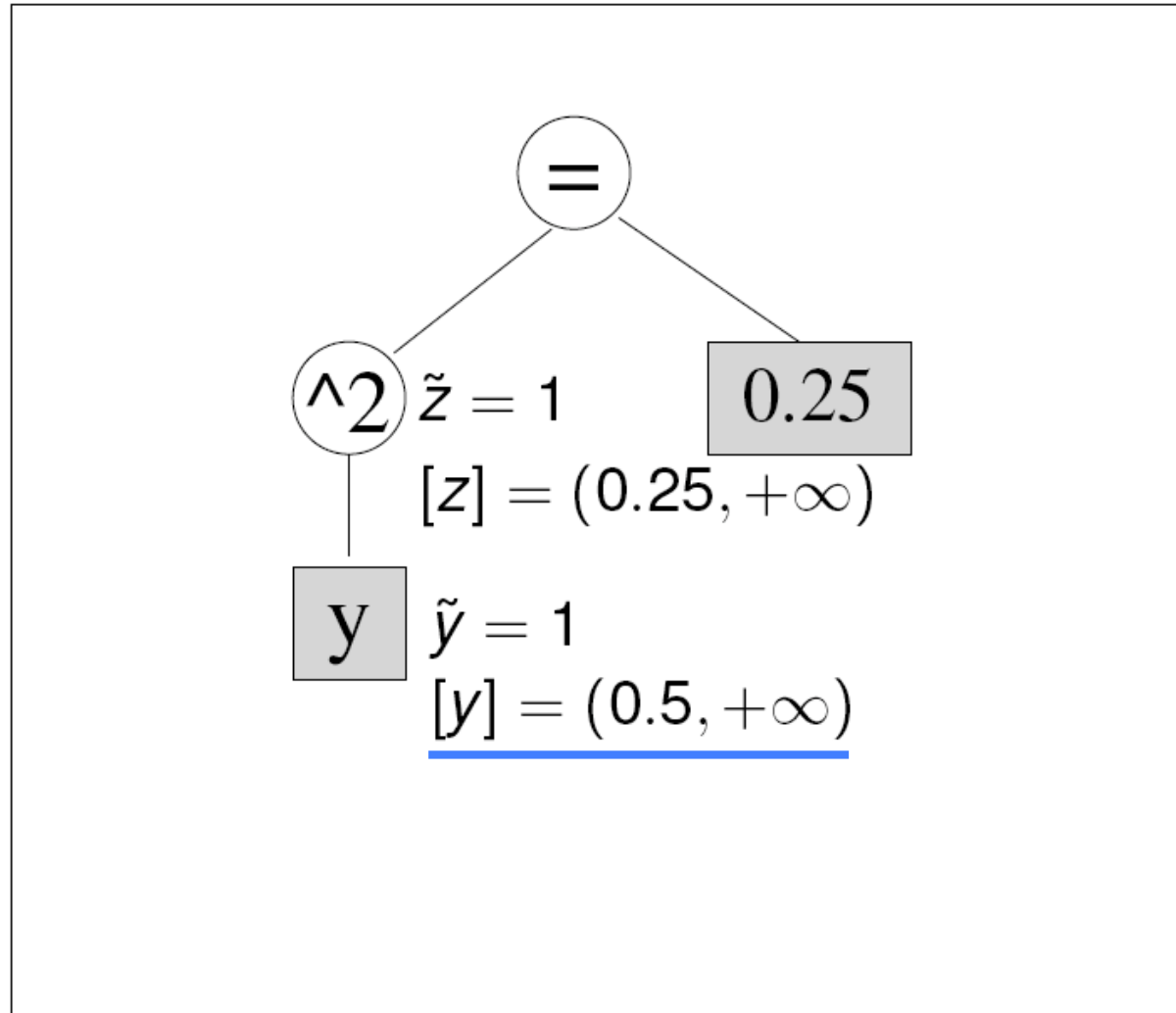
Two intervals satisfy $[y]^2 \subset [z]$

Main Algorithm (example)

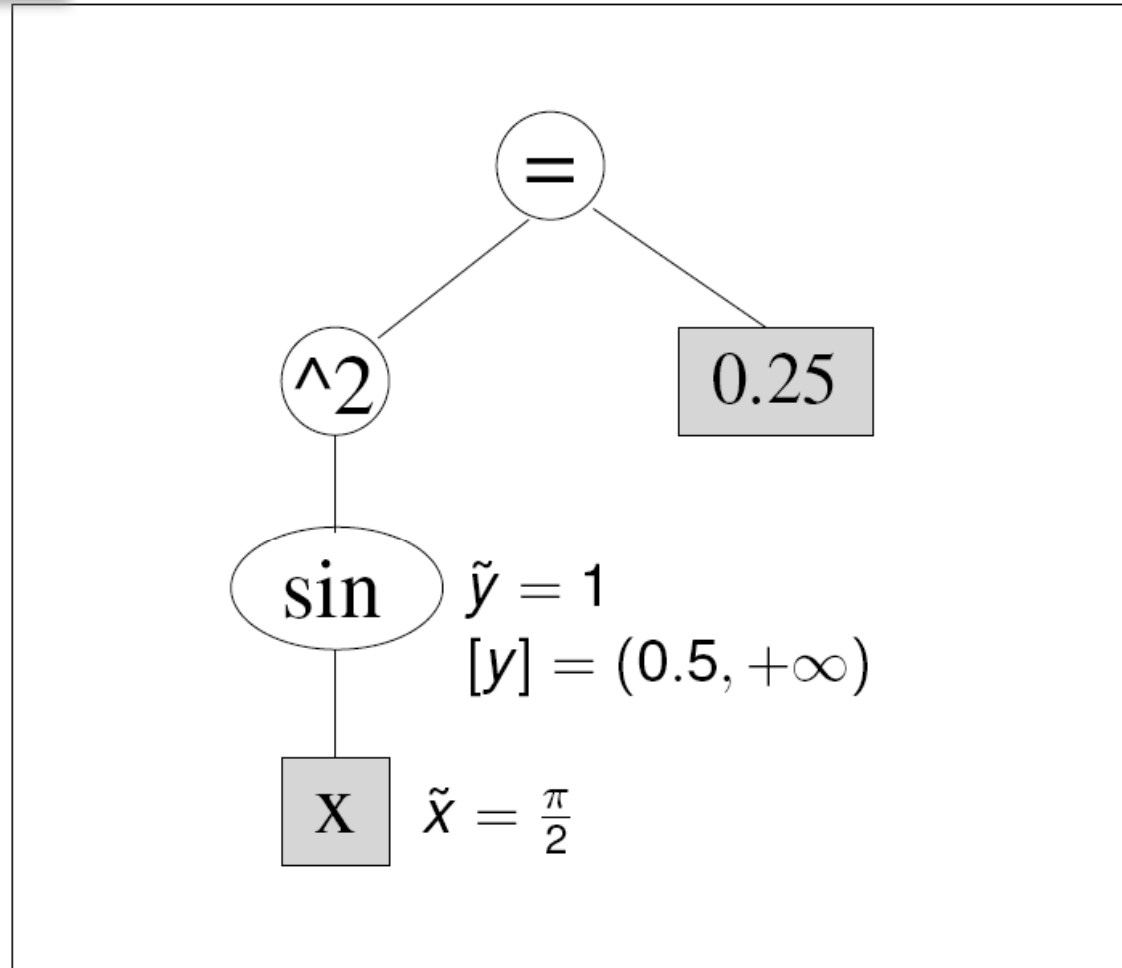


Select the one that contain \tilde{y}

Main Algorithm (example)

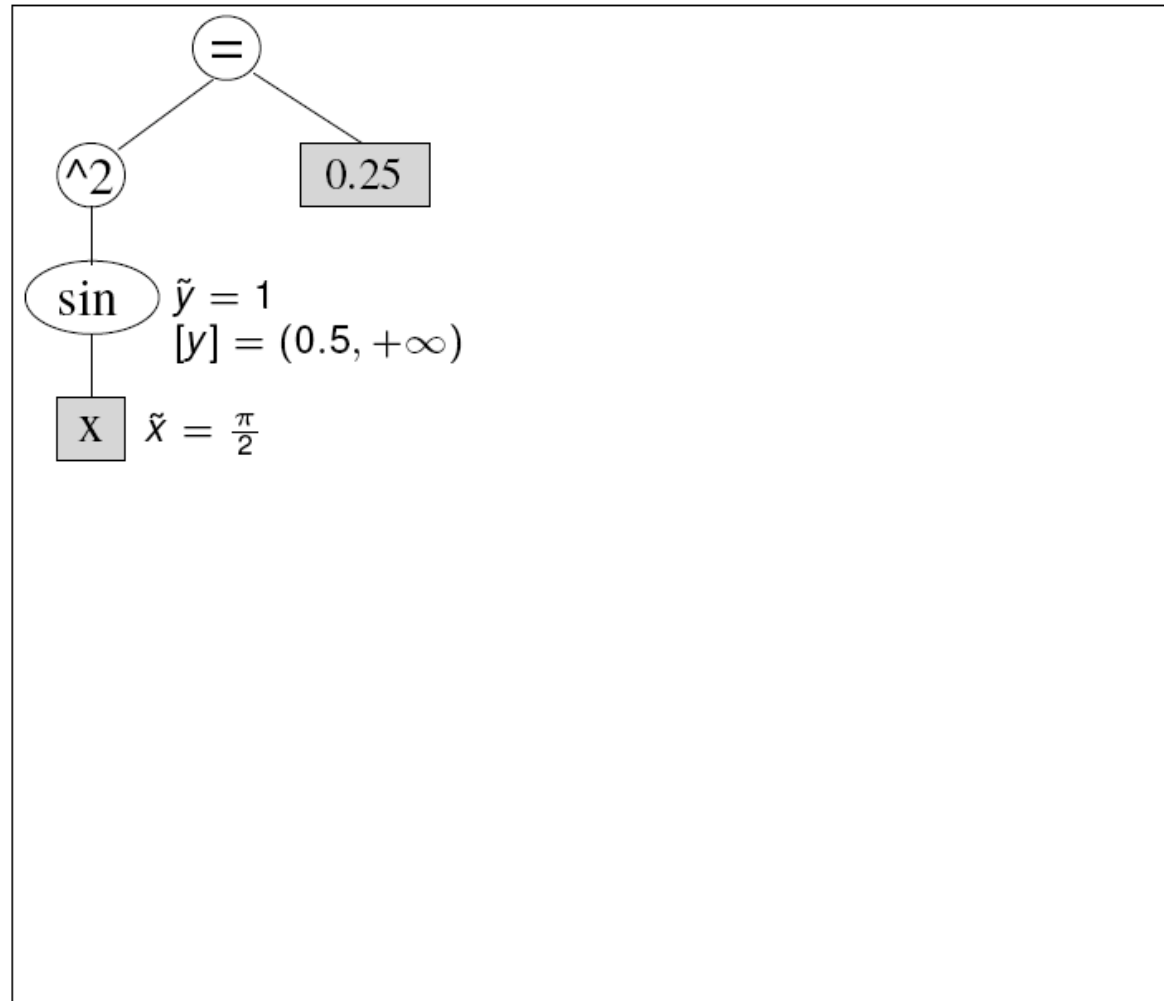


Main Algorithm (example)



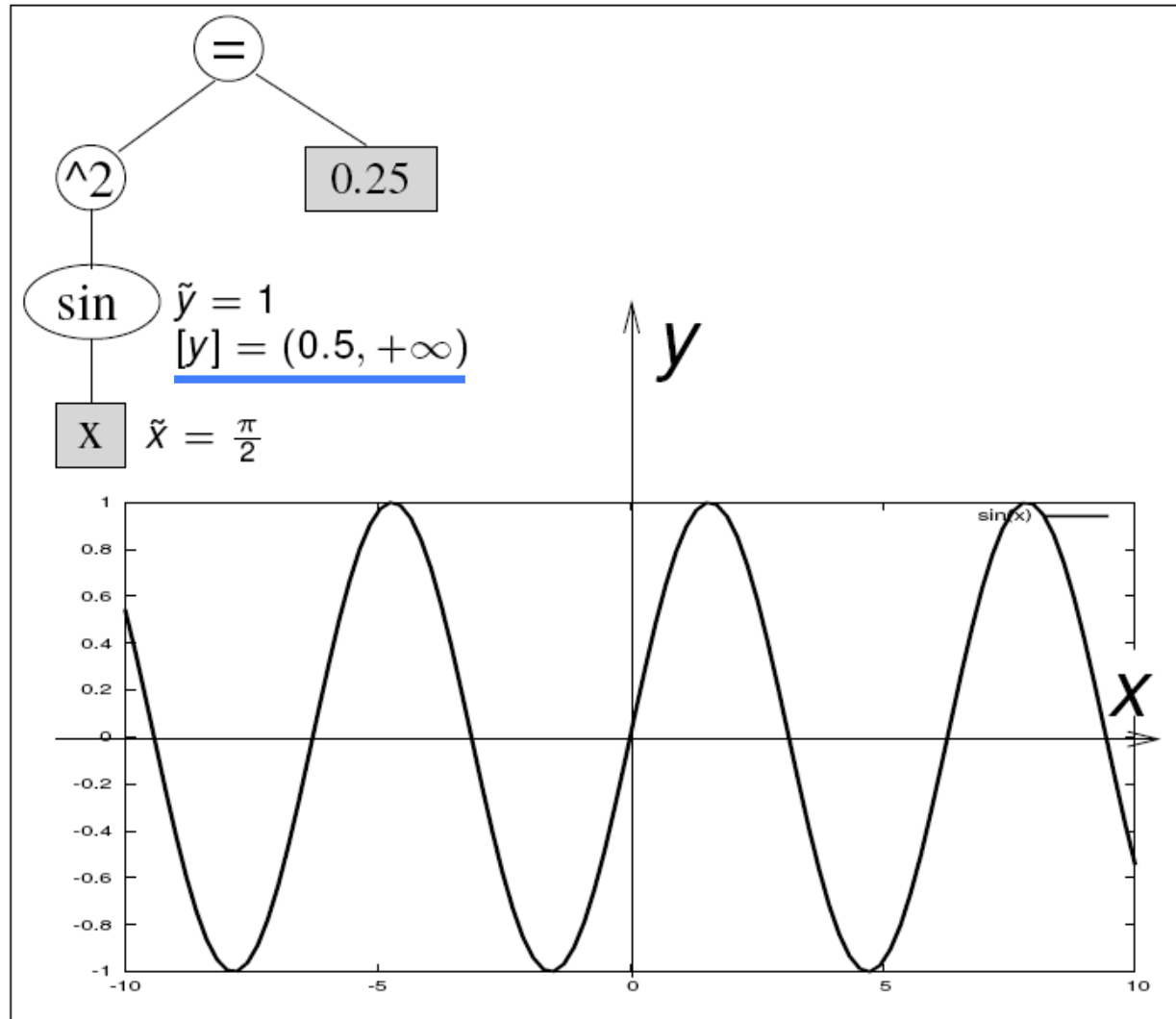
“Unfold” the node y

Main Algorithm (example)

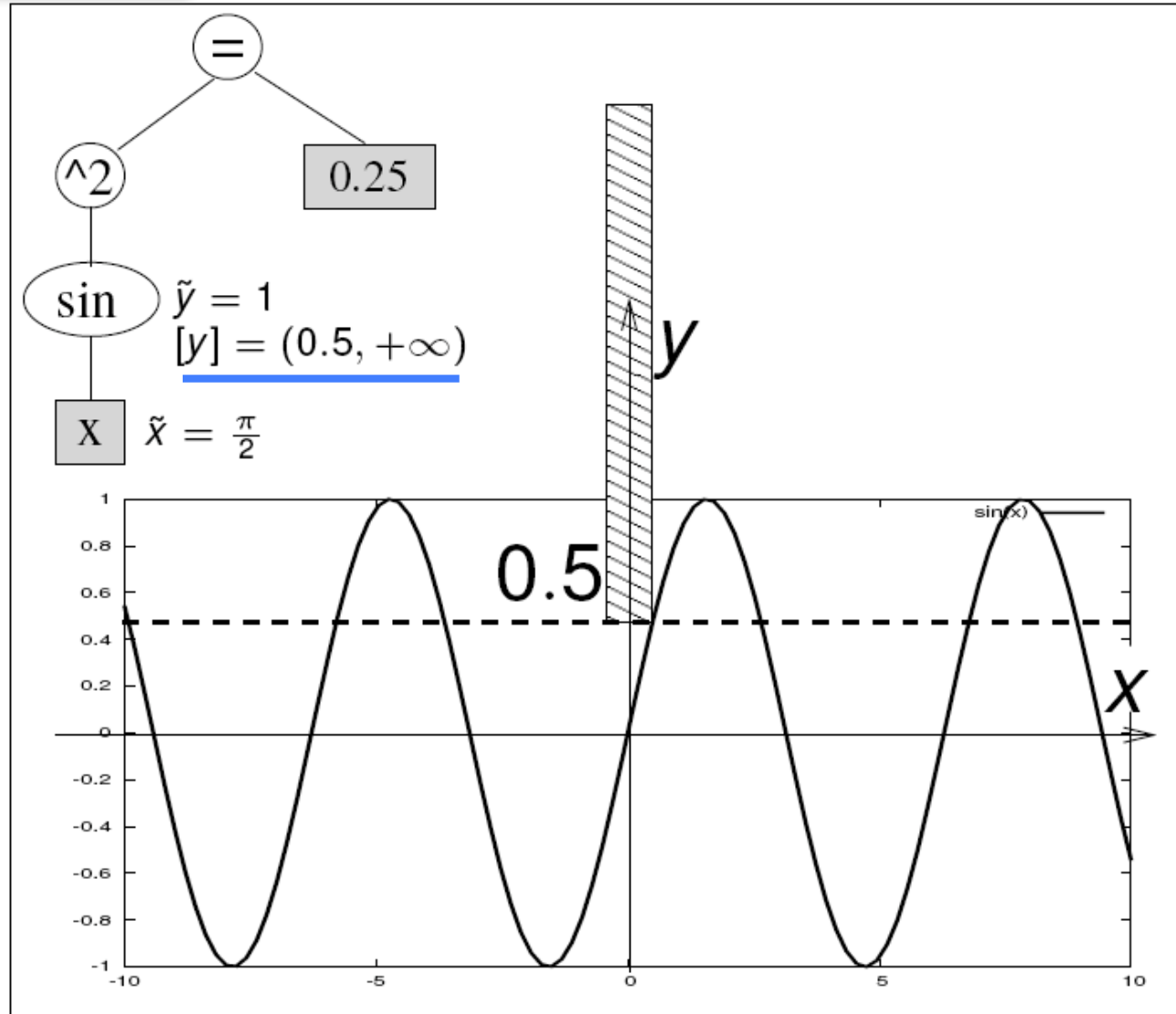


Repeat the procedure down to the leaf x

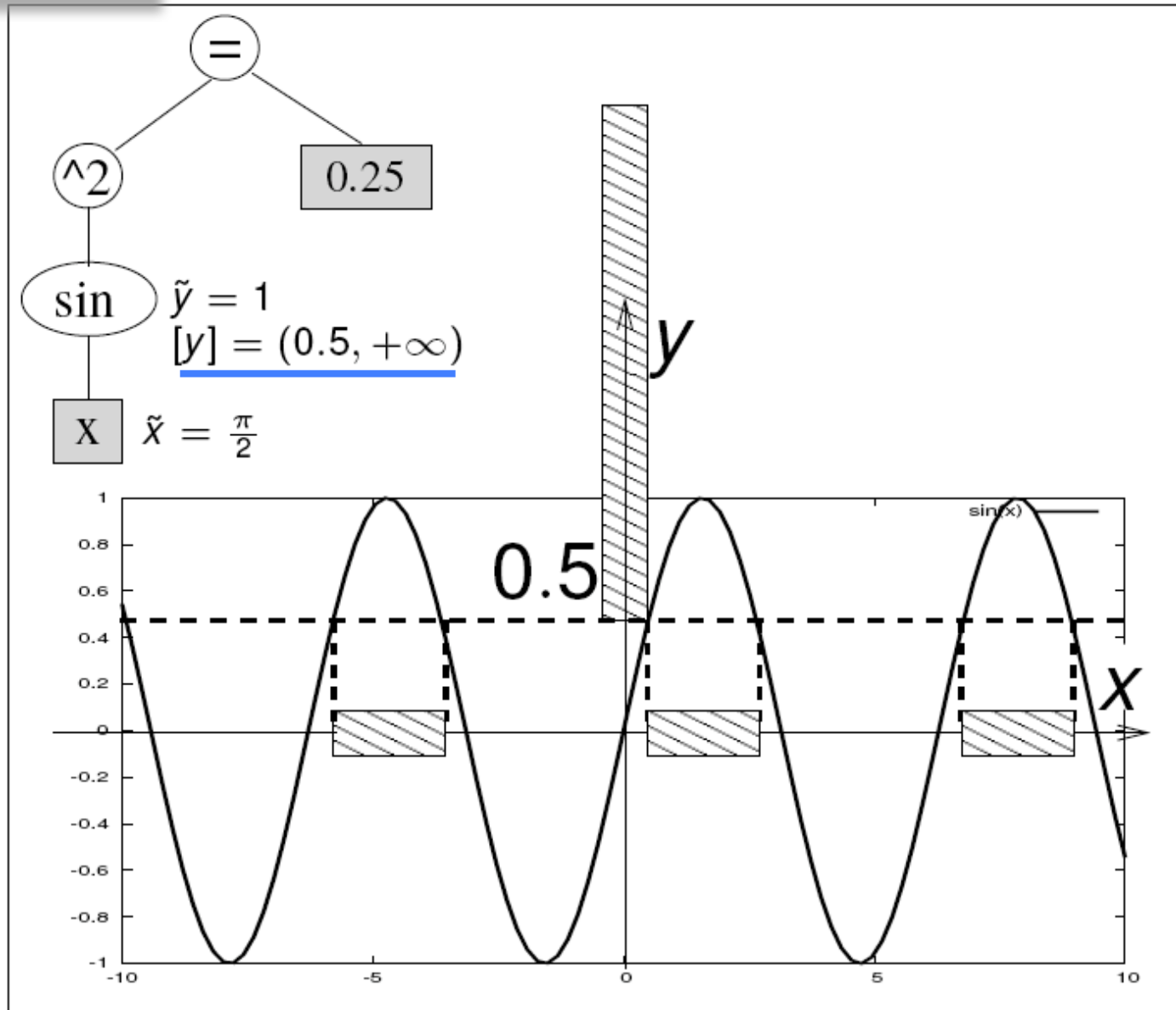
Main Algorithm (example)



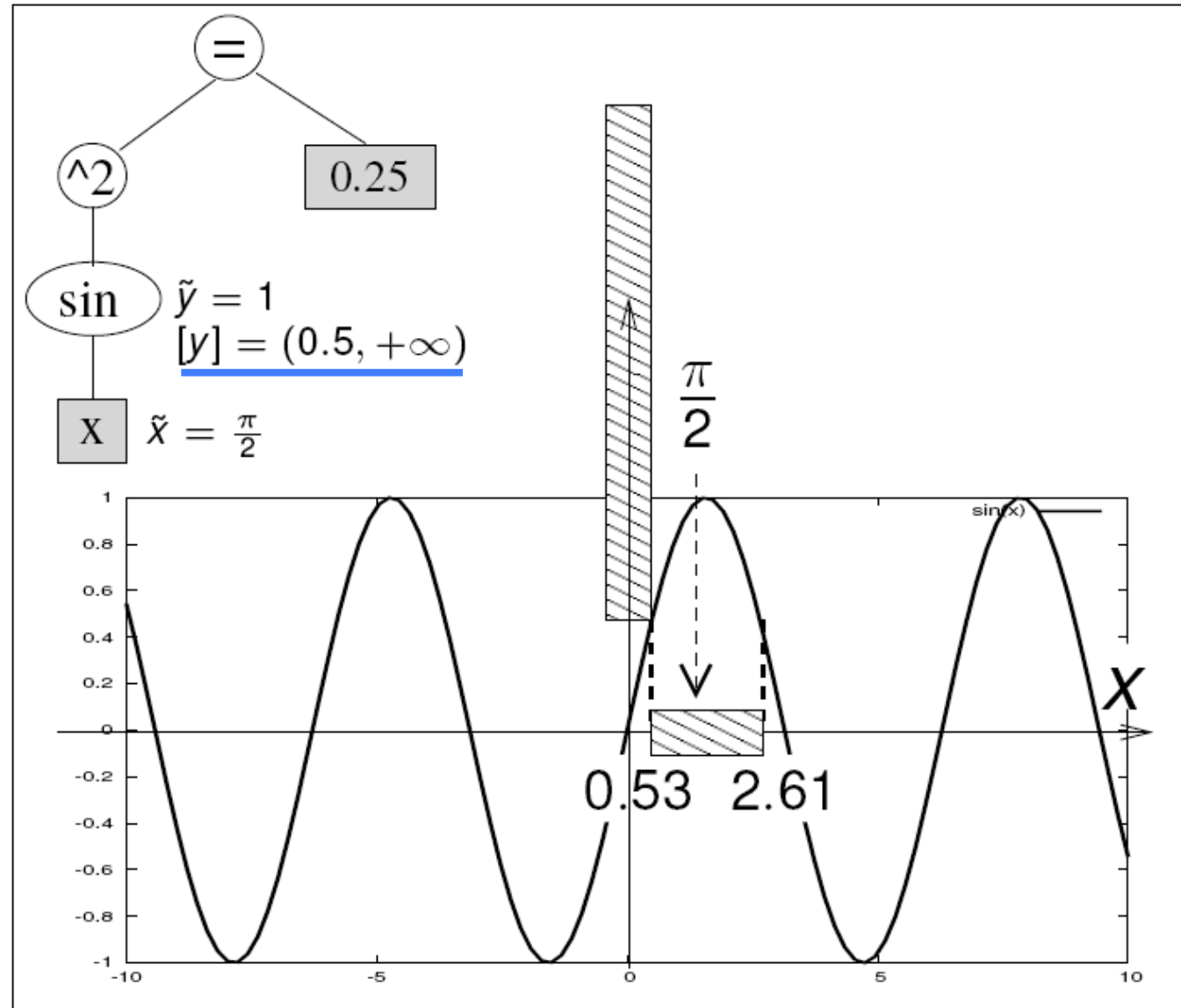
Main Algorithm (example)



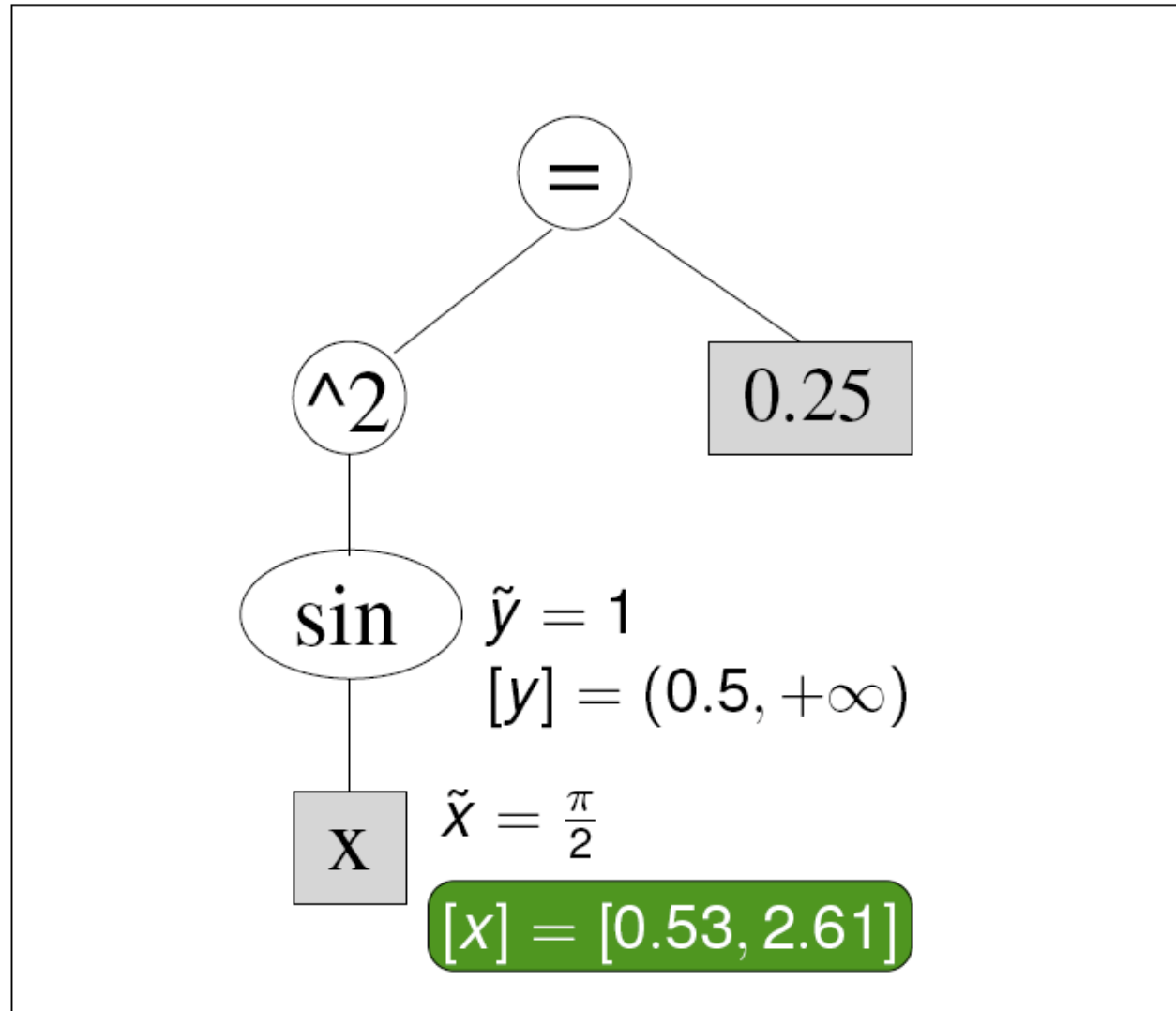
Main Algorithm (example)



Main Algorithm (example)



Main Algorithm (example)



CONTEXT AND MOTIVATION

FILTERING WITH SWEEP

A GENERIC INFLATER FOR ARITHMETICAL CONSTRAINTS

 **CONCLUSION**

Conclusion

Contribution

A **generic inflater** for any constraint on continuous domains that has a mathematical expression using $+$, \times , $-$, $/$ operators and functions (sqrt, sin, ...).

Complexity is **linear** in the length of the constraint expression.

Did not take multiple occurrences of a same variable:

This may current lead to underestimate the forbidden box we compute.

(multiple occurrences of a same variable can model extra alignment constraints)

There is still an engineering work to do to use this in an efficient way

(constant matters and may influence implementation)