Satisfiability Checking

Interval Constraint Propagation

Prof. Dr. Erika Ábrahám

RWTH Aachen University
Informatik 2
LuFG Theory of Hybrid Systems

WS 14/15

Motivation

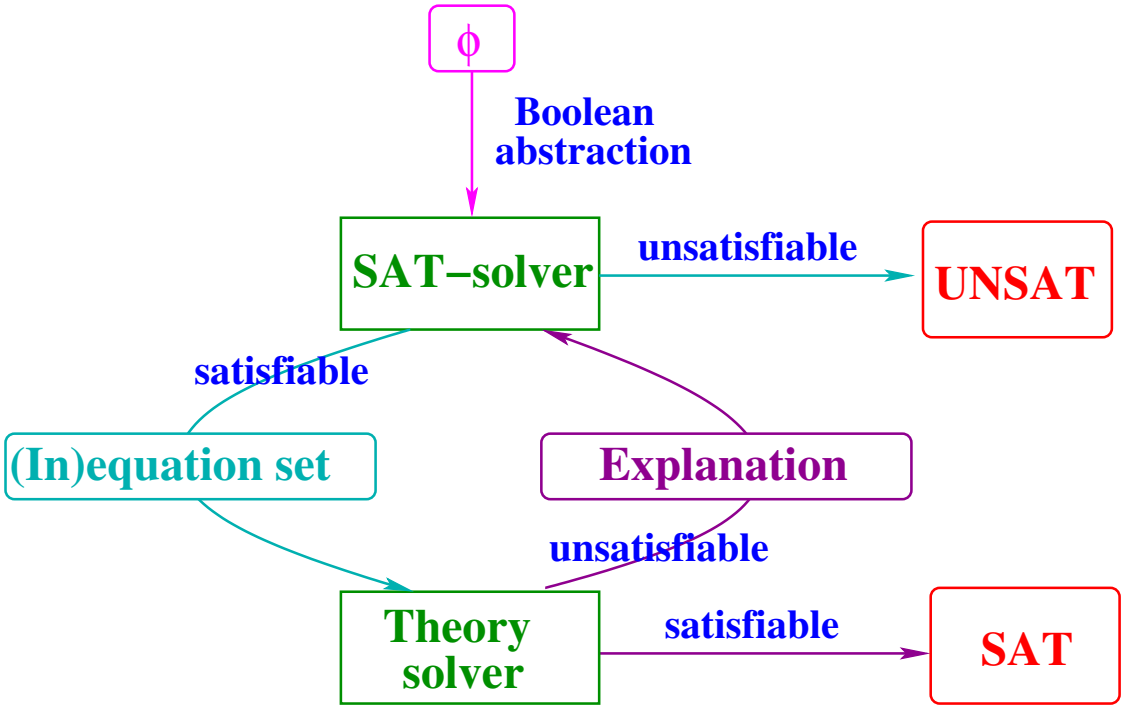
We consider input formulae φ from the theory of *quantifier-free nonlinear real arithmetic* (QFNRA):

$$p := r \mid x \mid (p + p) \mid (p - p) \mid (p \cdot p)$$
$$c := p < 0 \mid p = 0$$
$$\varphi := c \mid (\varphi \wedge \varphi) \mid \neg\varphi$$

$r \in \mathbb{Q}, x \in \mathit{Var}(\varphi)$

- Solving QFNRA has exponential complexity $\rightarrow$ hard to solve
- Current approaches for solving QFNRA:
  - Virtual substitution (VS)
  - Cylindrical algebraic decomposition (CAD)
  - Gröbner bases
  - ICP
- Procedures for linear real arithmetic (LRA) solving: Simplex

ICP



Interval constraint propagation [HMK97]

- Incomplete method
- Cheap reduction of search space
- Requires initial box

Interval

Definition (Interval)

Interval $I = [l_I; u_I] = \{x \in \mathbb{R} | l_I \sim x \sim u_I\}, l_I, u_I \in \mathbb{R}$, where l_I is called the lower bound and u_I the upper bound of I.

Possible bound types:

- Weak ($l_I \leq x \leq u_I$)
- Strict ($l_I < x < u_I$)
- Infinity ($-\infty < x < +\infty$)
- Combinations

Definition (Interval box)

N-dimensional box $B \in \mathbb{I}^n$.

Basic Interval arithmetic

Goal: Extend real arithmetic to intervals [Kul08].

- Coefficients can be intervals
- Operations maintain all correct solutions

Definition (Interval addition)

$A, B \in \mathbb{I} : A + B = [l_a + l_b; u_a + u_b]$

Example (Addition)

$[-1; 5] + [1; 4] = [0; 9]$

$[-2; 3] + 4 = [-2; 3] + \underbrace{[4; 4]}_{[4]} = [2; 7]$

Basic Interval arithmetic

Definition (Interval subtraction)

$A, B \in \mathbb{I} : A - B = [l_a - u_b; u_a - l_b]$

Example (Subtraction)

$[-1; 5] - [1; 4] = [-5; 4]$
$[-2; 3] - 4 = [-2; 3] + [4] = [-6; -1]$

Definition (Interval multiplication)

$A, B \in \mathbb{I} : A \cdot B =$
$[min(l_a \cdot l_b, l_a \cdot u_b, u_a \cdot l_b, u_a \cdot u_b); max(l_a \cdot l_b, l_a \cdot u_b, u_a \cdot l_b, u_a \cdot u_b)]$

Example (Multiplication)

$[-1; 5] \cdot [1; 4] = [-4; 20]$
$[-2; 3] \cdot 4 = [-2; 3] + [4] = [-8; 12]$

Basic Interval arithmetic

Special case of multiplication: Square

Definition (Interval square)

Squaring an interval can only result in positive values:
$A \in \mathbb{I} : A^2 = (A \cdot A) \cap [0; +\infty)$

Example (Square)

$[-1; 5]^2 = [0; 25]$

Definition (Interval division ($0 \notin B$))

$A, B \in \mathbb{I} : A \div B = A \cdot \frac{1}{B} = A \cdot [\frac{1}{u_b}; \frac{1}{l_b}]$

Example (Division)

$[2; 3] \div [4; 5] = [2; 3] \cdot \frac{1}{B} = [2; 3] \cdot [\frac{1}{5}; \frac{1}{4}] = [\frac{2}{5}; \frac{3}{4}]$

Extended Interval arithmetic

Problem: Division (divisor interval may contain 0):

Example (Division by zero)

$[1;3] \div [-2;3] = ?$
Usual approach:
$[1;3] \div [-2;3] = [1;3] \cdot [\frac{1}{3}; -\frac{1}{2}] \rightarrow$ invalid bounds (let's ignore this)
$[1;3] \cdot [\frac{1}{-2}; \frac{1}{3}] = [-\frac{3}{2}; 1]$ but: $\underbrace{\frac{1}{2}}_{\in[1;3]} \div \underbrace{\frac{1}{8}}_{\in[-2;3]} = \underbrace{4}_{\notin[-\frac{3}{2};1]}$

Introduce new rules for extended interval division [Kul08] such that:
$[1;3]/[-2;3] = (-\infty; \frac{1}{-2}] \cup [\frac{1}{3}; +\infty)$

Propagation

Propagation of intervals can imply new bounds

- Insert current bounds for all except one variable in one constraint
- Apply interval arithmetic to gain new bounds
- Update bounds via intersection

Example (Propagation)

$x \in [1;3], y \in [1,2], c_1 : y = x, c_2 : y = x^2$
$c_2, x : x = \sqrt{y} \rightarrow x = \sqrt{[1;2]} = [1;\sqrt{2}] \rightarrow x \in [1;3] \cap [1;\sqrt{2}] = [1;\sqrt{2}]$
$c_1, y : y = x \rightarrow y = [1;\sqrt{2}] \rightarrow y \in [1;2] \cap [1;\sqrt{2}] = [1;\sqrt{2}]$

Contraction sequence:
$x : [1;2] \overset{c_2,x}{\rightarrow} [1;\sqrt{2}] \overset{c_2,x}{\rightarrow} [1;\sqrt[4]{2}] \overset{c_2,x}{\rightarrow} [1;\sqrt[8]{2}] \overset{c_2,x}{\rightarrow} \ldots \overset{c_2,x}{\rightarrow} [1;1]$
$y : [1;3] \overset{c_1,y}{\rightarrow} [1;\sqrt{2}] \overset{c_1,y}{\rightarrow} [1;\sqrt[4]{2}] \overset{c_1,y}{\rightarrow} [1;\sqrt[8]{2}] \overset{c_1,y}{\rightarrow} \ldots \overset{c_1,y}{\rightarrow} [1;1]$

We define a pair of constraint and variable as a contraction candidate (CC).

Preprocessing

Propagation requires constraints to be solvable in one variable
$\rightarrow$ Preprocessing required: Linearization.

- Separate linear and nonlinear constraints
- Introduce new variables for nonlinear monomials
- Obtain two sets: Linear constraints (including linearizations) and nonlinear monomials

Preprocessing

Toy example:

$$x^2 \cdot y + z = 0$$

$$v_1 + z = 0 \wedge v_1 = x^2 \cdot y$$

Algorithm

Setup:

- Input: Conjunction of constraints
- Variables bounded by intervals (initial box)

Goal

Reduce the size of the input box to a specified diameter while guaranteeing that a solution (if available) is still contained inside the box.

Note: In this context we pass the gained box (solution candidate) to a backend implementing a complete method.

- We stop contraction when some diameter is reached
- Complete methods profit from reduced search space

Algorithm

```
Answer icp(Box init, ConstraintSet constraints):
  Box current = init;
  bool finished = false;
  while(!finished){
    bool done = false;
    while(!done) {
      CC candidate = chooseCandidate(constraints);
      done = contract(current, candidate); // CBR: current
    }
    if(!empty(current))
      if(callBackend(current) == SAT)
        return SAT;
    else
      finished = chooseBox(current);
  }
  return UNSAT;
```

Algorithm

Setup:

- Conjunction of constraints
- Variables bounded by intervals (initial box)
- Goal: Reduce box for complete methods (backend)

Requirements:

- Method to choose CCs
- Method for contraction
- Mechanism to keep track of boxes
- Mechanism to choose next box

Contraction

General approach: Contract via propagation.
Problems:

- Contraction is in general not predictable
- Contraction may stop before target diameter reached
- Contraction may cause a split (heteronomous split)

Example (Contraction candidate choice)

Consider $\{c_1 : y = x, c_2 : y = x^2\}$ with initial intervals $I_x := [1, 3]$ and $I_y := [1, 2]$
At each step we can consider 4 contractions:

- $I_x \overset{c_1, x}{\rightarrow} [1, 2]$ (33% reduction)
- $I_y \overset{c_1, y}{\rightarrow} [1, 2]$ (0% reduction)
- $I_x \overset{c_2, x}{\rightarrow} [1, \sqrt{2}]$ (47% reduction)
- $I_y \overset{c_2, y}{\rightarrow} [1, 2]$ (0% reduction)

$\rightarrow$ Contraction gain varies.

Reinforced learning

We can improve the choice of CCs by heuristics:

- The algorithm selects the next contraction candidate with the highest weight $W_k^{(ij)} \in [0; 1]$.
- Afterwards the weight is updated (according to the relative contraction $r_{k+1}^{(ij)} \in [0; 1]$).

Weight updating:

$$W_{k+1}^{(ij)} = W_k^{(ij)} + \alpha(r_{k+1}^{(ij)} - W_k^{(ij)})$$

The factor $\alpha \in [0; 1]$ decides how the importance of the events is rated:

- Large α (e.g. 0.9) $\rightarrow$ The last recent event is most important
- Small α (e.g. 0.1) $\rightarrow$ The initial weight is most important

CCs with a weight less than some threshold ε are not considered for contraction.

Contraction

Requirements for contraction operator:

- Does not drop solutions
- Detects if no solution is contained in the current box

The second usually cannot be guaranteed.

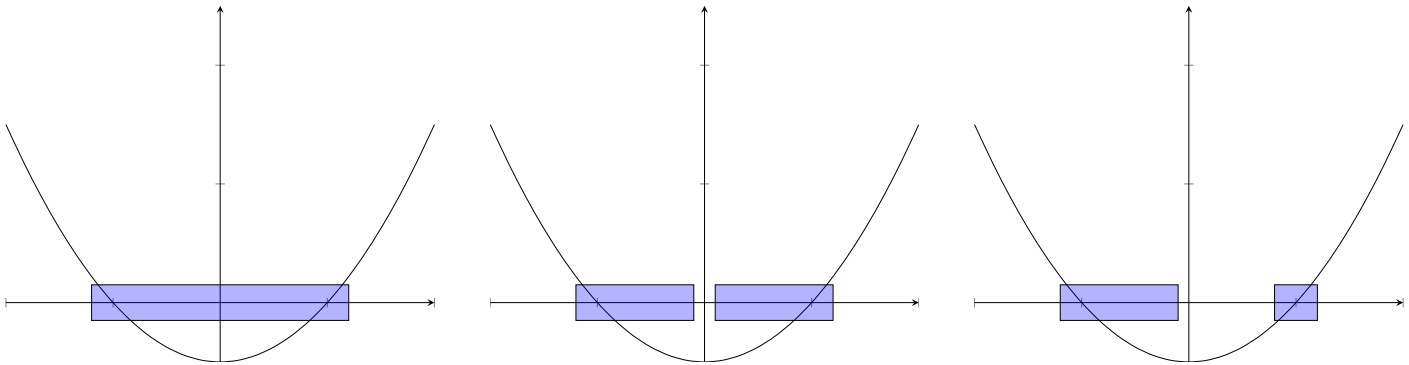
Usual contraction operators:

- Propagation (original contractor)
- Interval Newton method

General approach: Contract as long as contraction gain is large enough.

Splitting

When the weight of all CCs is below the threshold we do not make progress $\rightarrow$ split manually (autonomous split).



Problems:

- How to store boxes
- How to select the next box

History Tree

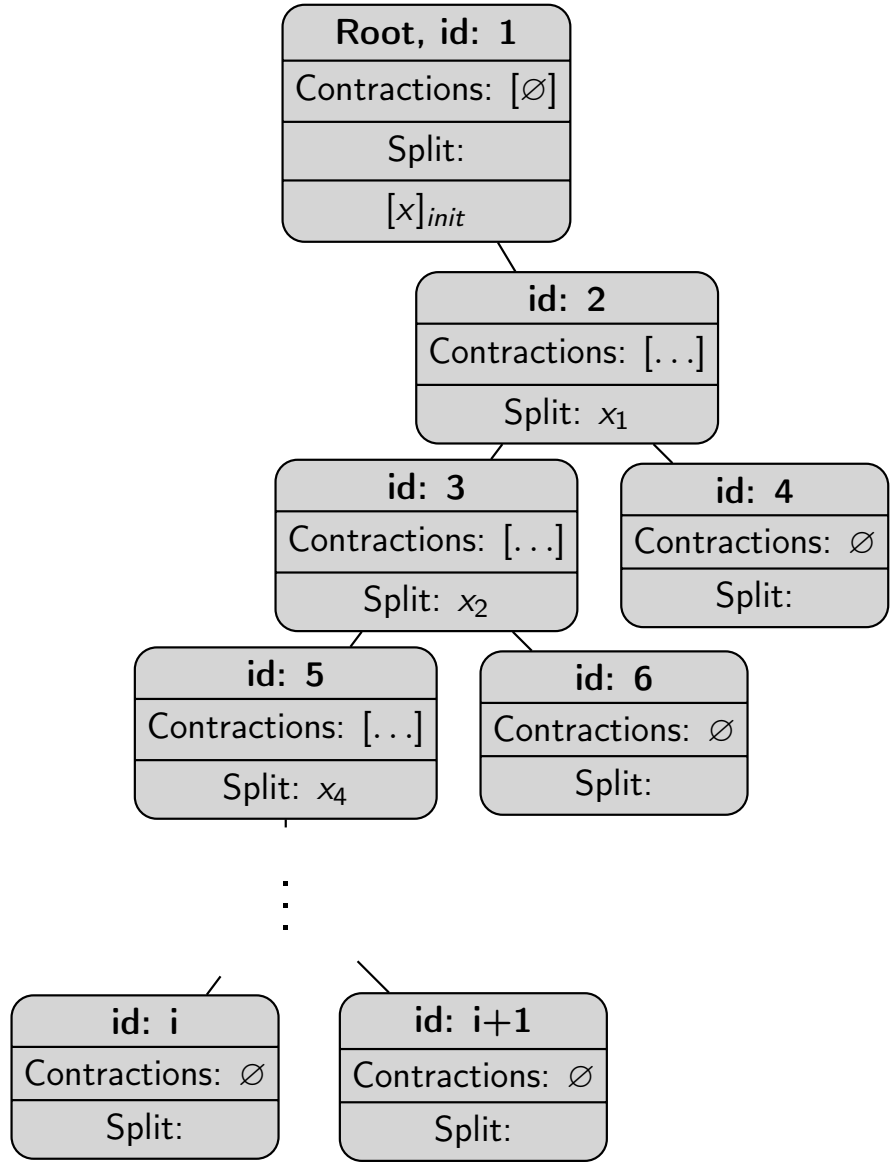
To keep track of current status we utilize a tree-structure, which holds solver states:

- Search box
- Applied contractions
- Splitting dimension

Additionally we can use the tree to collect infeasible subsets:

1. Infeasible box $\rightarrow$ propagate reasons to parent
2. (optional) skip boxes

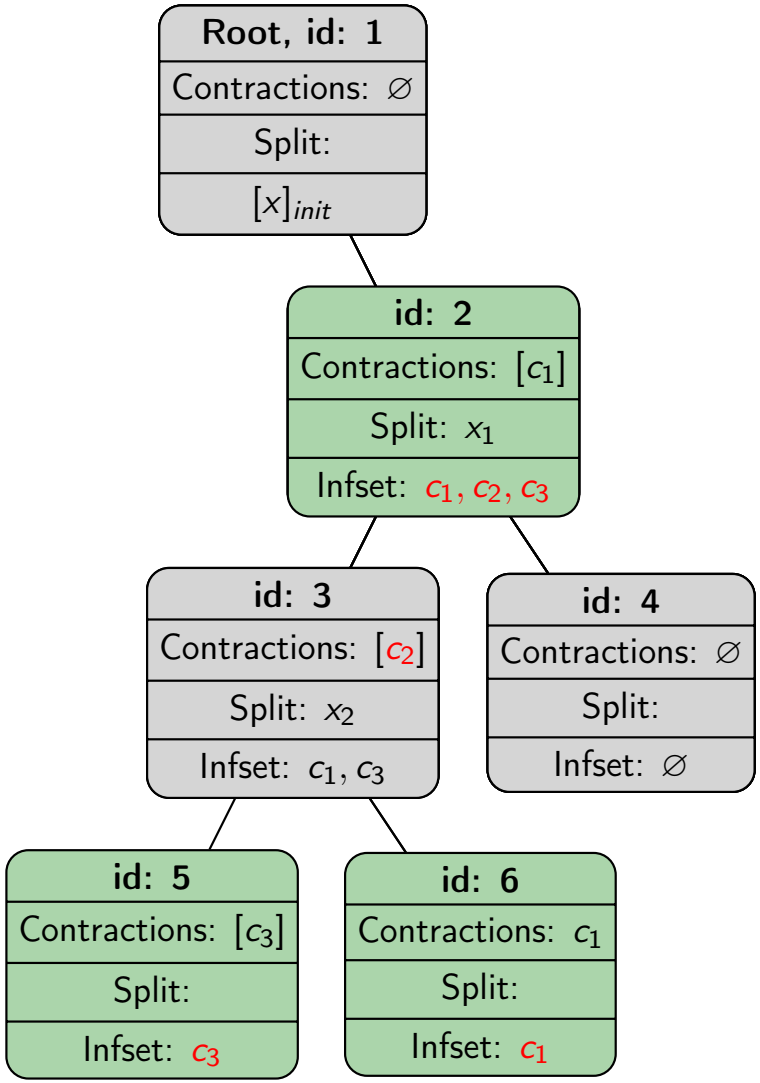
$\rightarrow$ We generate a set of constraints which includes the infeasible subset.



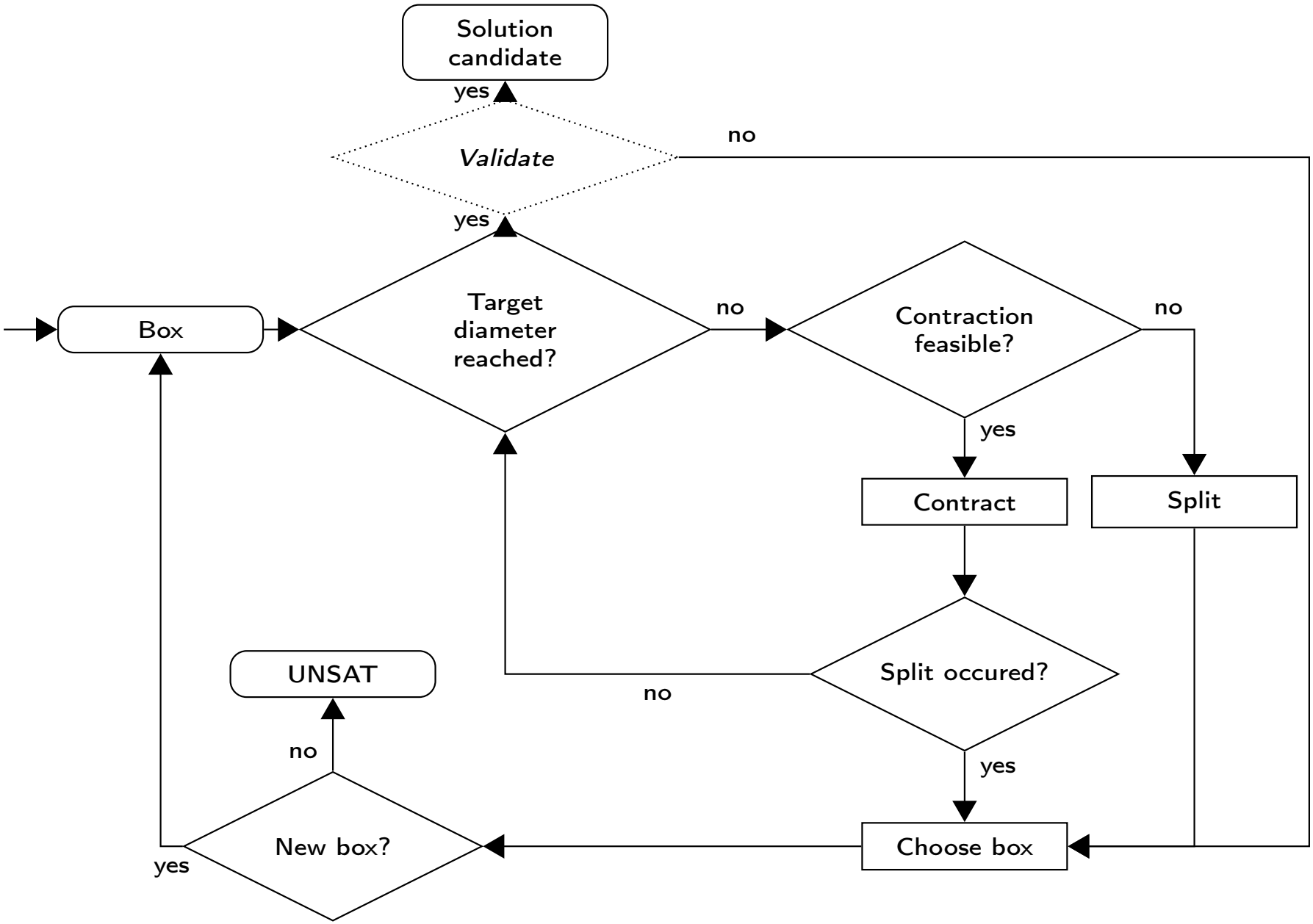
If no further heuristics is applied we traverse the tree pre-order.

History tree example

Setup: constraints $c_1, \ldots, c_4$, variables $x_1, \ldots, x_3$



Algorithm overview



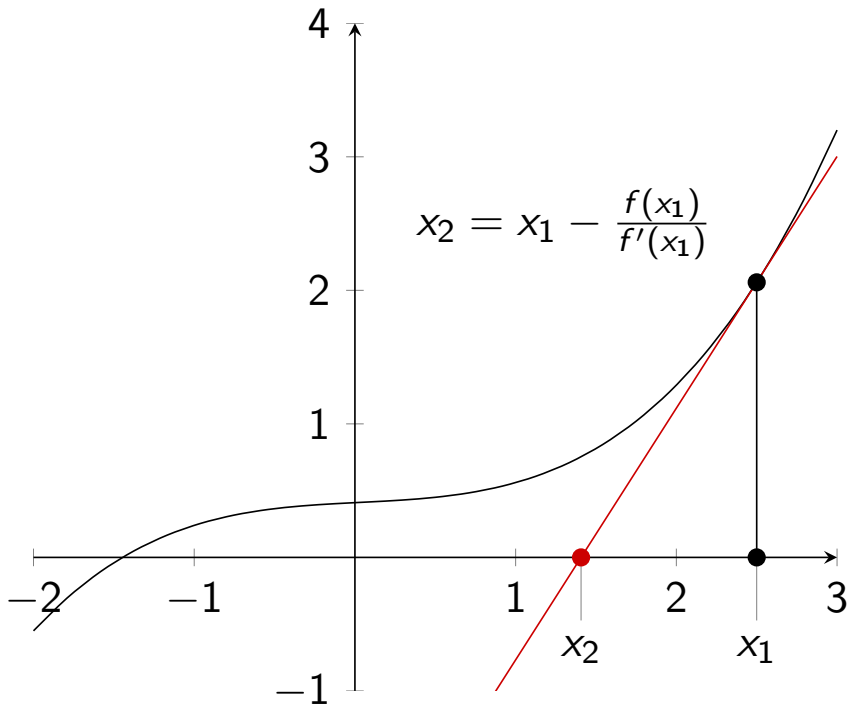
Improvements

Additional improvements:

- More sophisticated contraction (Interval Newton method [MKC09])
- Use linear solver for linear constraints
- Introduce box validation
- Involve SAT-Solver for box-choice
- Introduce splitting heuristics

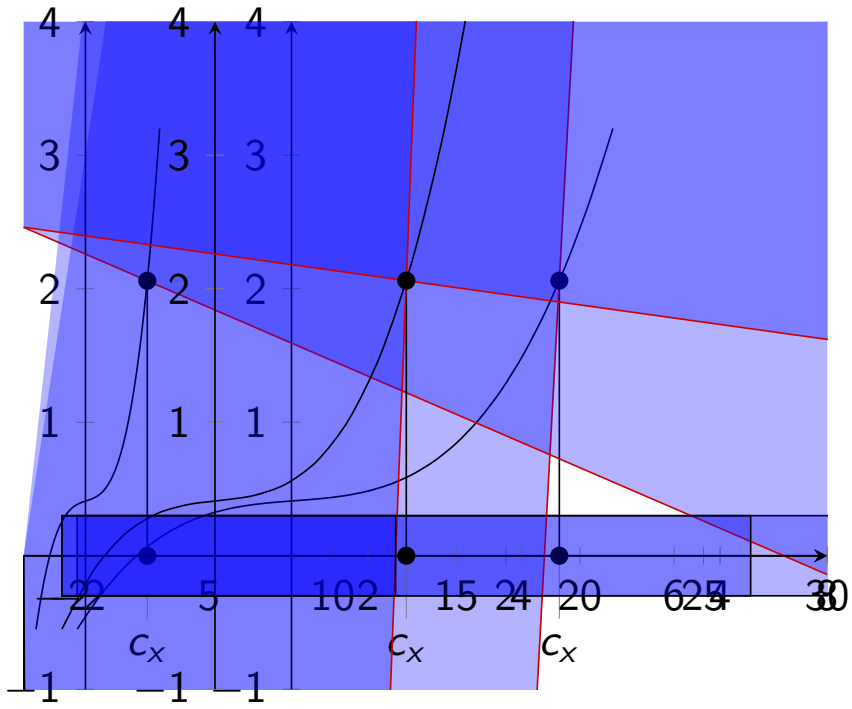
Interval Newton method

Reminder: Newton method for root finding (univariate polynomials):



Interval Newton method

Interval extension of Newtons method [HR97]:



$f(x) = 0.1 \cdot x^3 - 0.01 \cdot (x-3)^2 + 0.5$ Starting interval: $x \in [-2; 7]$, sampling point: $c_x = 2.5$ Tangent:
$[2.5] - \frac{f(c_x)}{f'(x)} = (-\infty; 2.36018] \cup [28.25; +\infty)$ New interval:
$x \in [-2; 7] \cap (-\infty; 2.36018] \cup [28.25; +\infty) = [-2; 2.36018]$

Interval Newton method

Componentwise Newton operator

$$N_{cmp}(\overbrace{[x]}^{box}, \overbrace{f_i(x_1,\ldots,x_n), j}^{CC}) :=$$
$$c([x]_{x_j}) - \frac{f_i([x]_{x_1},\ldots,[x]_{x_{j-1}}, c([x]_{x_j}), [x]_{x_{j+1}},\ldots,[x]_{x_n})}{\frac{\partial f_i}{\partial x_j}([x]_{x_1},\ldots,[x]_{x_n})}$$

Reminder(Multivariate Newton):

$N_{cmp}(x, f_i(x_1,\ldots,x_n), j) = x - \frac{f_i(x)}{\frac{\partial f_i}{\partial x_j}(x)}$ The operator N_{cmp} has two

important properties:

- If x^* is a solution and $x^* \in [x]$, then $x^* \in N_{cmp}([x], i, j)$
- If $[x] \cap N_{cmp}([x], i, j) = \varnothing$, there is no solution in $[x]$

$\rightarrow$ Advantage: No diverging behavior like the original Newton method due

Introduce linear solver

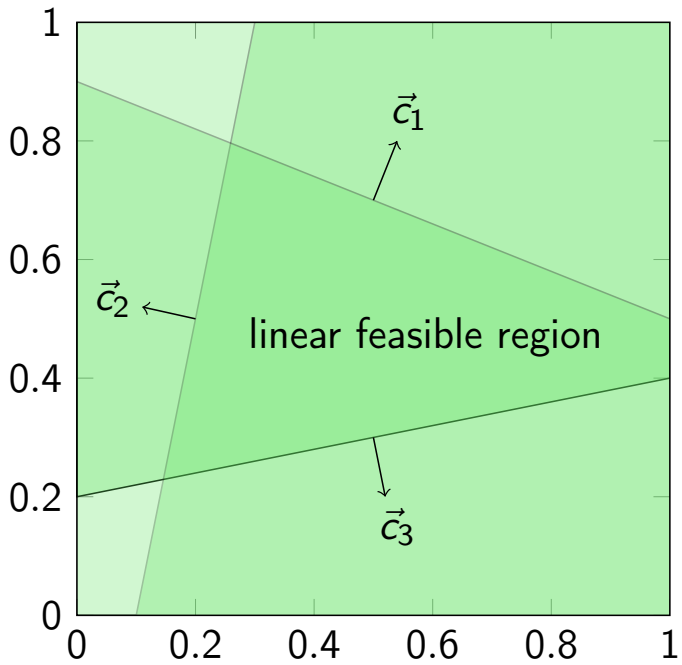
ICP is not well-suited for linear problems
($\rightarrow$ slow convergence phenomenon).

Make use of linear solvers for linear constraints:

- Pre-process to separate linear and nonlinear constraints
- Determine linear feasible region
- Use nonlinear constraints for contraction
- Validate resulting boxes against linear feasible region
- In case box is linear infeasible: Add violated linear constraint for contraction

Validation

- Resulting intervals represent n–dimensional hyperboxes
- Boxes may violate linear constraints
- Check resulting boxes against the linear feasible region

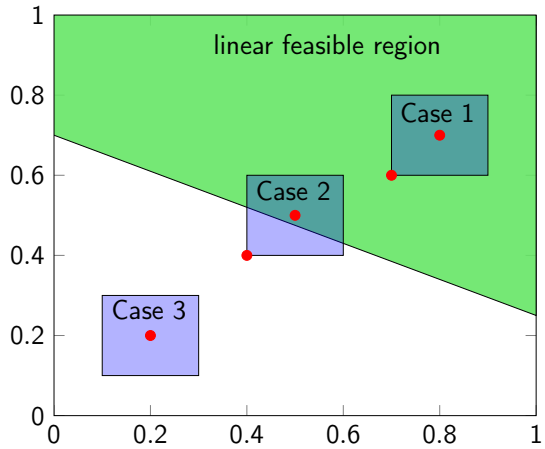


Case distinction

We have to distinct 3 cases:

- Case 1: The resulting interval lies completely inside the linear feasible region
- Case 2: The hyperbox partly covers the linear feasible region
- Case 3: The solution interval lies outside the linear feasible region

The problem lies in separating the 1st case from the 2nd one.



If a checked point is valid $\rightarrow$ return SAT

Raise splitting

Instead of having an internal box management (e.g. via a tree) we can raise a split to the SAT solver

- Create deductions (tautologies)
- Deduction is added as additional information to the SAT solver
- SAT solver decides based on its information which split to take

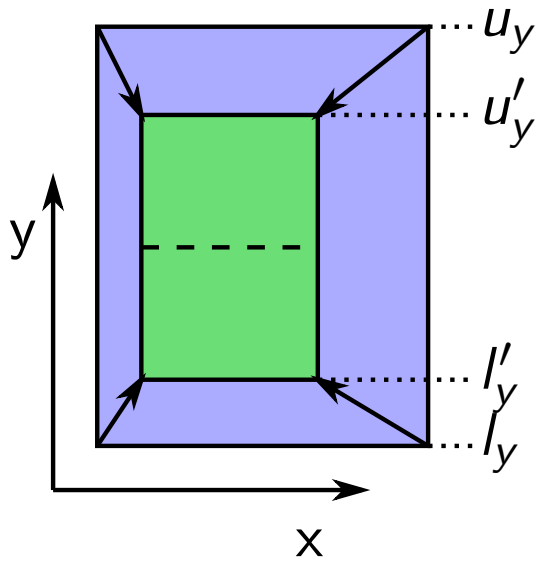
ICP deductions include:

- The split
- (optional) A premise in form of previous contractions

Raise splitting

Setup:

- Original box (blue) contracted to new box (green)
- Contraction via constraints $c_i, i = 1, \ldots, n$
- Demanded split at $c(l_y)$



We create a splitting deduction:

$$\underbrace{x \in [l_x] \wedge y \in [l_y] \wedge (c_1 \wedge \ldots \wedge c_n)}_{\textit{premise}} \rightarrow \underbrace{(y \in [l'_y; c(l_y)] \oplus y \in (c(l_y); u'_y])}_{\textit{split}}$$

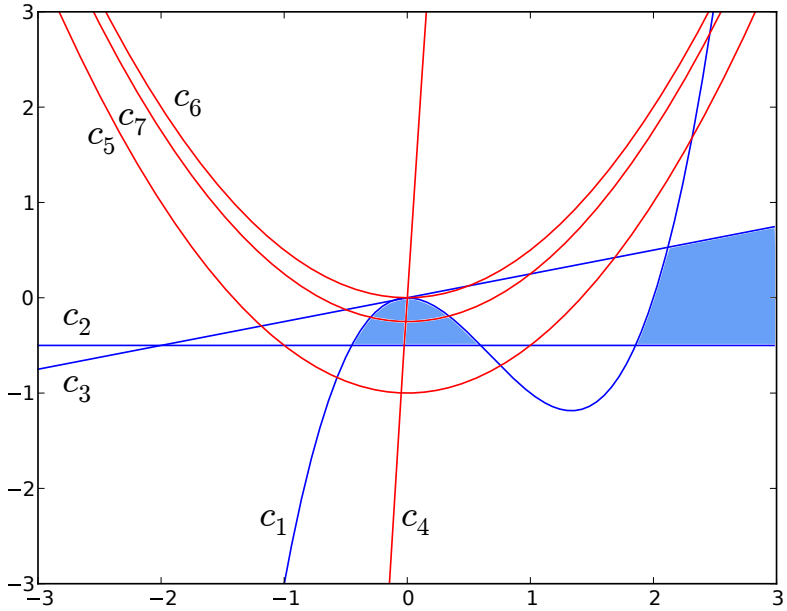
Experimental results

Results of a prototype implementation:

- Linear and nonlinear constraints
- Weak and strict inequations

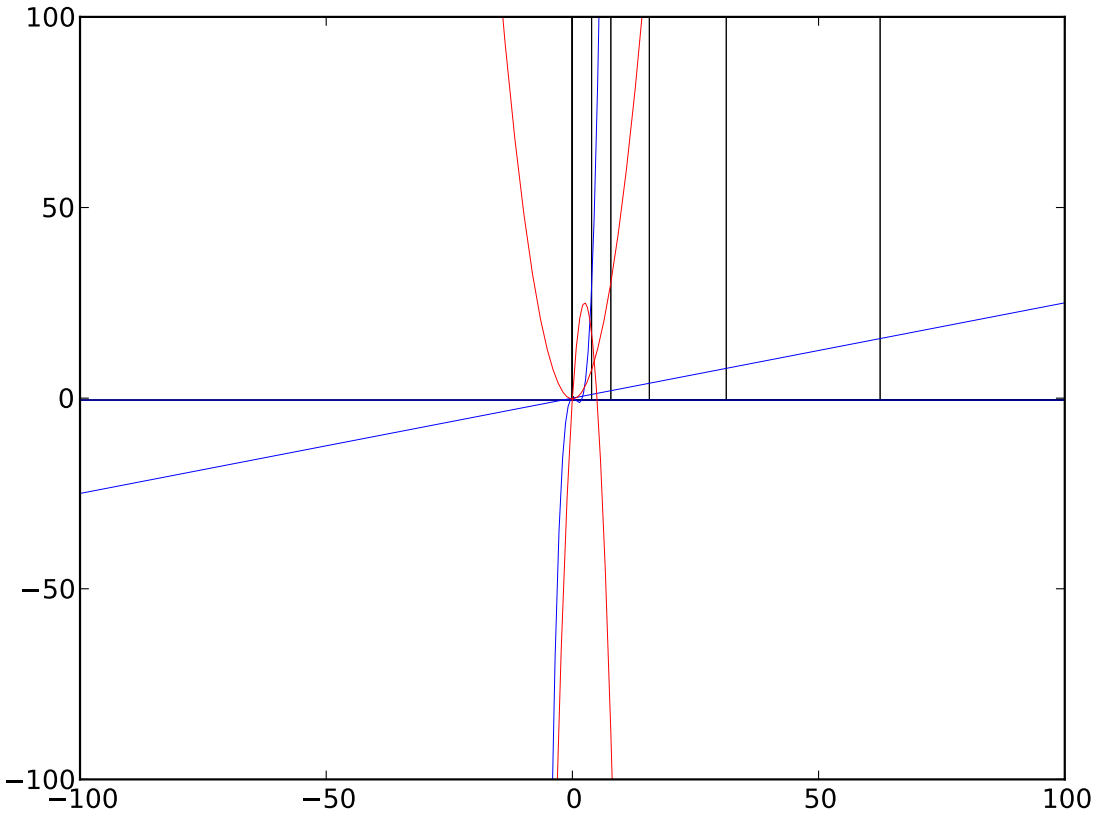
Test example:

$$\left[\left(c_1 : x^3 - 2x^2 > y \right) \right.$$
$$\wedge \left(c_2 : -\frac{1}{2} \leq y \right)$$
$$\wedge \left(c_3 : \frac{1}{4}x \geq y \right)$$
$$\wedge \left(c_4 : -4x^2 + 20x = y \vee c_5 : \frac{1}{2}x^2 - 1 = y \right)$$
$$\left. \wedge \left(c_6 : x^2 = y \vee c_7 : \frac{1}{2}x^2 - \frac{1}{4} = y \right) \right]$$



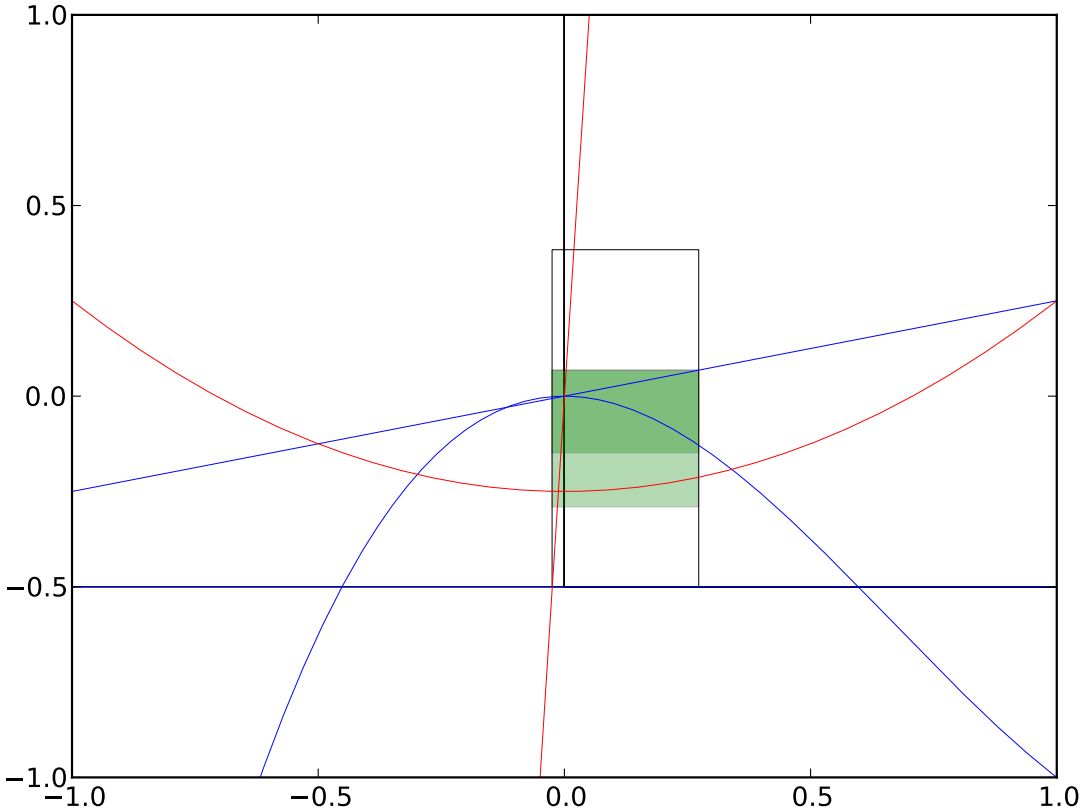
Experimental results

target size = 1



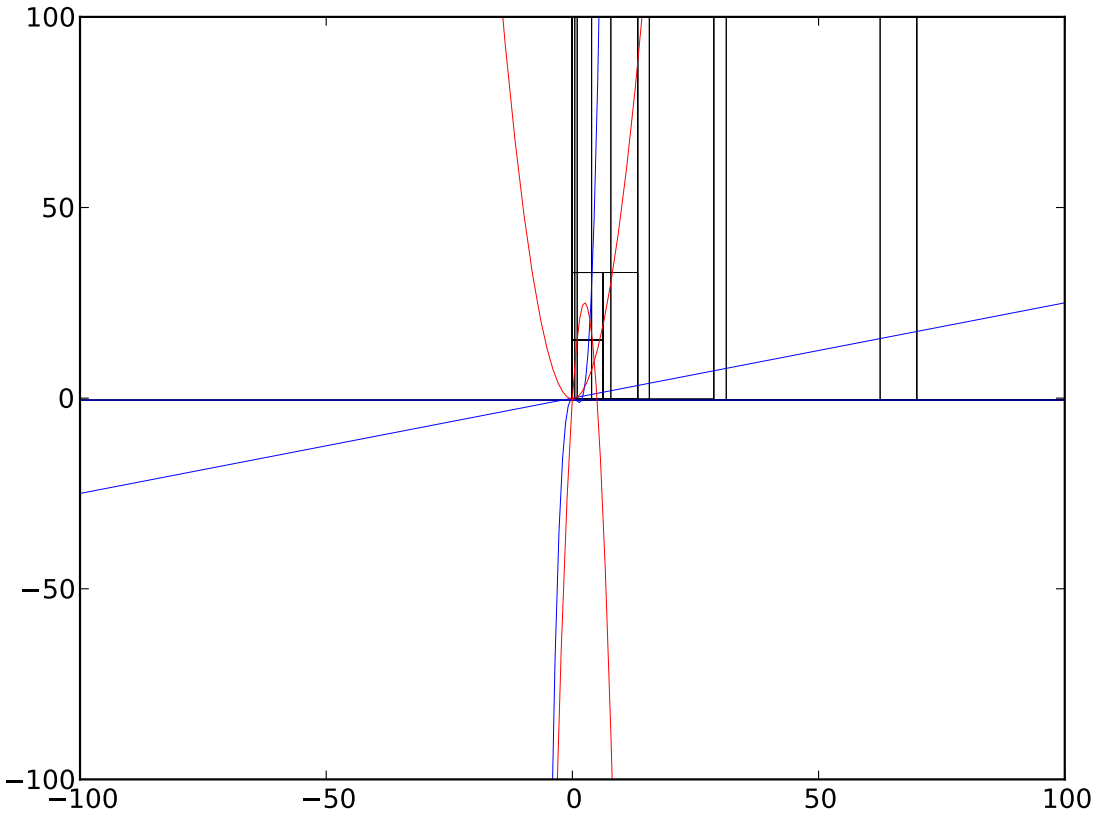
Experimental results

target size = 1



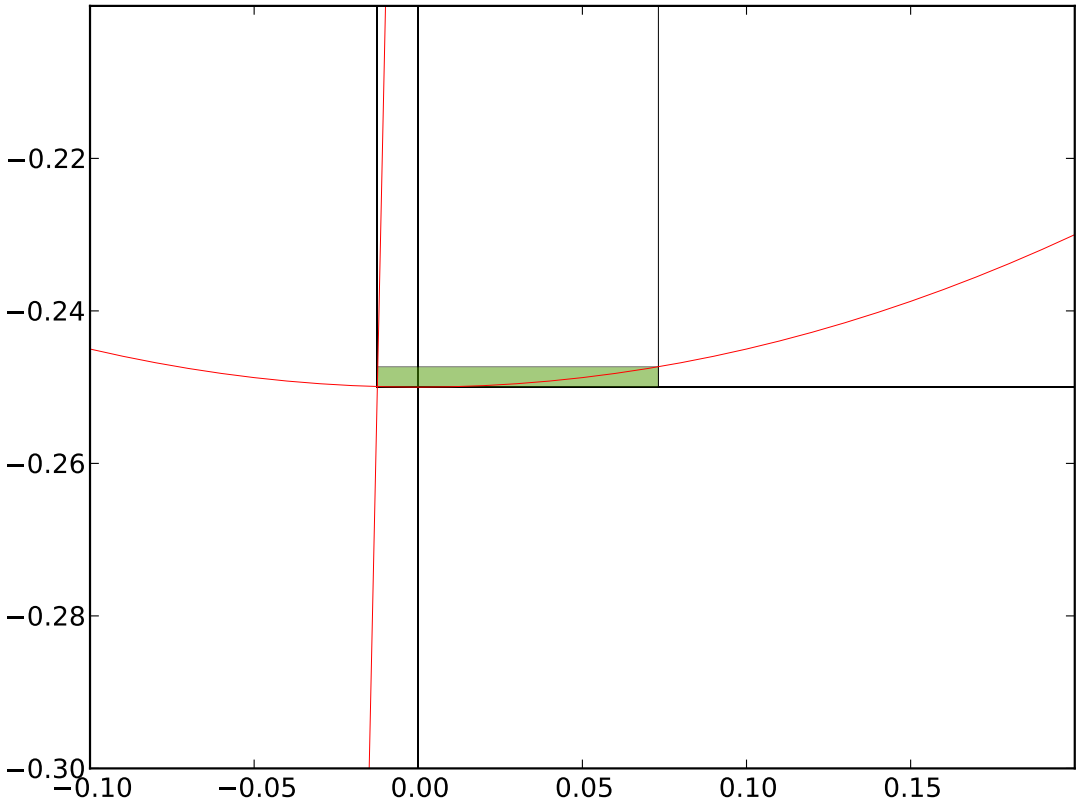
Experimental results

target size = 0.1



Experimental results

target size = 0.1



Experimental results

∅	backend calls	inv. boxes	depth avg(max)	# contr.	# splits
10	8	1	28(64)	155	100
1	7	2	25(66)	219	120
0.1	4	5	36(82)	369	177
0.01	7	9	39(94)	525	218
0.001	4	17	43(106)	638	247
0.0001	4	22	48(122)	735	281

Summary

Aspects relevant for exercise and exam:

- Basic interval arithmetic (no division by intervals containing 0)
- Interval propagation
- Problems during contraction
- Splitting
- Basic algorithm (see Slide 20)
- Possible extensions (not in detail)

Bibliography I


P. Van Hentenryck, D. Mcallester, and D. Kapur.
Solving polynomial systems using a branch and prune approach.
SIAM Journal on Numerical Analysis, 34:797–827, 1997.


Stefan Herbort and Dietmar Ratz.
Improving the efficiency of a nonlinear-system-solver using a componentwise Newton method.
Forschungsschwerpunkt Computerarithmetik, Intervallrechnung und numerische Algorithmen mit Ergebnisverifikation. Institut für angewandte Mathematik Karlsruhe, 1997.


Ulrich Kulisch.
Complete interval arithmetic and its implementation on the computer.
Numerical Validation in Current Hardware Architectures, Dagstuhl Seminar Proceedings(08021), 2008.

Bibliography II


R. Moore, R. Kearfott, and M. Cloud.
Introduction to interval analysis.
Society for industrial and applied mathematics, 2009.