Satisfiability Checking

Interval Constraint Propagation

Prof. Dr. Erika Ábrahám

RWTH Aachen University
Informatik 2
LuFG Theory of Hybrid Systems

WS 16/17

Non-linear real arithmetic

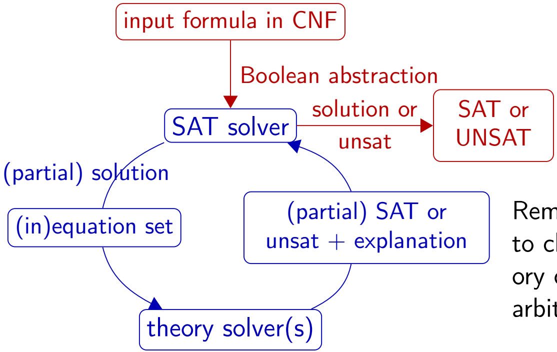
We consider input formulae φ from the theory of quantifier-free nonlinear real arithmetic (QFNRA):

$$
\begin{array}{lll}
p & := \mathit{const} \mid x \mid (p + p) \mid (p - p) \mid (p \cdot p) & \text{polynomials} \\
c & := p < 0 \mid p = 0 & \text{(polynomial) constraints} \\
\varphi & := c \mid (\varphi \wedge \varphi) \mid \neg\varphi & \text{QFNRA formulas}
\end{array}
$$

where constants *const* and variables x take real values from $\mathbb{R}$.

- Best known methods for checking the satisfiability of QFNRA formulas have exponential complexity $\rightarrow$ hard to solve
- Approaches we learn for solving QFNRA:
  - Interval constraint propagation (ICP) incomplete
  - Virtual substitution (VS) incomplete
  - Cylindrical algebraic decomposition (CAD) complete

Interval constraint propagation (ICP) in SMT



Remember: the theory solvers needs to check sets/conjunctions of theory constraints only (in contrast to arbitrary Boolean combinations)

We first use interval constraint propagation (ICP) in a theory solver module:

- Incomplete: ICP always terminates but it might return "unknown" → we extend it with a backend implementing a complete procedure.
- Relatively cheap reduction of the search space: Even if the answer is "unknown", ICP might still be helpful because it returns a smaller search space (a set of subsets of the original search space) without loosing any solution.

Intervals

In the following we consider closed intervals only ($\mathbb{R}$ denotes the real numbers).

Definition (Interval)

An interval $A = [\underline{A}, \overline{A}] = \{v \in \mathbb{R} \mid \underline{A} \leq v \leq \overline{A}\}$ is a closed and connected subset of $\mathbb{R}$, defined by its

- lower bound $\underline{A} \in \mathbb{R} \cup \{-\infty\}$ and its
- upper bound $\overline{A} \in \mathbb{R} \cup \{+\infty\}$,

where $-\infty \leq v \leq +\infty$ for all real numbers $v \in \mathbb{R}$.
We call A bounded if both of its bounds are real-valued ($\underline{A} \neq -\infty$ and $\overline{A} \neq +\infty$), and unbounded otherwise. Let $\mathbb{I}$ be the set of all intervals.

For point intervals $[v, v] = \{v\}$ for some $v \in \mathbb{R}$ we also write v.

Intervals and boxes

Definition (Interval diameter)

The width/diameter D_A of an interval $A = [\underline{A}, \overline{A}] \in \mathbb{I}$ is $D_A = +\infty$ if A is unbounded and $D_A = \overline{A} - \underline{A}$ otherwise.

Q: What is the width of a point interval?
A: 0

Q: If we know the width of an interval, how can we determine whether the interval is empty?
A: An interval is empty iff its width is negative.

Definition (Interval box)

An *n-dimensional box* is a cross product $B = B_1 \times \ldots \times B_n \in \mathbb{I}^n$ of n intervals.

Interval arithmetic

First we extend real arithmetic operations to intervals. Besides the interval-adaptations $+, -, \cdot : \mathbb{I} \times \mathbb{I} \to \mathbb{I}$ of the QFNRA operators $+, -, \cdot : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, we will also need division $\div : \mathbb{I} \times \mathbb{I} \to \mathbb{I}$ as the inverse of the multiplication, and square and square root operations $^2, \pm\sqrt{\ } : \mathbb{I} \to \mathbb{I}$ (we will see later why).

- Constants and variables are now interval-valued
- Interval operations are conservatively over-approximating with respect to their real-valued counterparts, i.e.,

  $$op\ A \supseteq \{\ op\ a \mid a \in A\}$$

  for $op \in \{\ ^2, \pm\sqrt{\ }\}$, and

  $$A\ op\ B \supseteq \{a\ op\ b \mid a \in A \wedge b \in B\}$$

  for $op \in \{+, -, \times, \div\}$.
- The approach introduced in this lecture can be naturally extended to further operators like *sin*, *cos*, *exp*, ...

Computing with infinity

We first partially extend the operations $+, -, \cdot, \div : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ from $\mathbb{R}$ to $\mathbb{R} \cup \{-\infty, +\infty\}$ as follows. Let $a, b \in \mathbb{R}$. The following tables define the extensions, where rows constain the first and columns the second operands.

Addition	$-\infty$	b	$+\infty$
$-\infty$	$-\infty$	$-\infty$	
a	$-\infty$	$a+b$	$+\infty$
$+\infty$		$+\infty$	$+\infty$

Subtraction	$-\infty$	b	$+\infty$
$-\infty$		$-\infty$	$-\infty$
a	$+\infty$	$a-b$	$-\infty$
$+\infty$	$+\infty$	$+\infty$	

Multiplication	$-\infty$	$b<0$	0	$b>0$	$+\infty$
$-\infty$	$+\infty$	$+\infty$	0	$-\infty$	$-\infty$
$a<0$	$+\infty$	$a \cdot b$	0	$a \cdot b$	$-\infty$
0	0	0	0	0	0
$a>0$	$-\infty$	$a \cdot b$	0	$a \cdot b$	$+\infty$
$+\infty$	$-\infty$	$-\infty$	0	$+\infty$	$+\infty$

Division	$-\infty$	$+\infty$
a	0	0

Note: The above tables define the arithmetic operations only partially (e.g., division $A \div B$ is not defined for $A \in \{-\infty, +\infty\}$). It is important to mention that the undefined cases (for which a meaningful definition cannot be given) will not be needed.

Interval arithmetic: Addition

Example (Interval addition)

$[-1;5] + [1;4] = [0;9]$
$[-2;3] + 4 = [-2;3] + \underbrace{[4;4]}_{4} = [2;7]$

Definition (Interval addition)

We define $A + B = [\underline{A} + \underline{B}; \overline{A} + \overline{B}]$ for all non-empty $A = [\underline{A}, \overline{A}] \in \mathbb{I}$ and $B = [\underline{B}, \overline{B}] \in \mathbb{I}$, and $A + B = \emptyset$ otherwise (if either A or B is empty).

Interval arithmetic: Subtraction

Example (Interval subtraction)

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

Definition (Interval subtraction)

We define $A - B = [\underline{A} - \overline{B}; \overline{A} - \underline{B}]$ for all non-empty $A = [\underline{A}, \overline{A}] \in \mathbb{I}$ and $B = [\underline{B}, \overline{B}] \in \mathbb{I}$, and $A - B = \emptyset$ otherwise.

We can also define unary minus as syntactic sugar:

Definition (Unary interval minus)

We define $-A = 0 - A = [-\overline{A}; -\underline{A}]$ for all $A = [\underline{A}, \overline{A}] \in \mathbb{I}$.

Interval arithmetic: Multiplication

Example (Interval multiplication)

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

Definition (Interval multiplication)

We define $A \cdot B =$
$[min(\underline{A} \cdot \underline{B}, \underline{A} \cdot \overline{B}, \overline{A} \cdot \underline{B}, \overline{A} \cdot \overline{B}); max(\underline{A} \cdot \underline{B}, \underline{A} \cdot \overline{B}, \overline{A} \cdot \underline{B}, \overline{A} \cdot \overline{B})]$ for all non-empty $A = [\underline{A}, \overline{A}] \in \mathbb{I}$ and $B = [\underline{B}, \overline{B}] \in \mathbb{I}$, and $A \cdot B = \emptyset$ if either A or B is empty.

Interval arithmetic: Multiplication

Example (Interval square)

Special case: Squaring an interval can only result in positive values.
$[-1;5]^2 = [0;25]$

Definition (Interval square)

We define $A^2 = (A \cdot A) \cap [0;+\infty)$ for all $A = [\underline{A}, \overline{A}] \in \mathbb{I}$.

Example (Interval square root)

$\pm\sqrt{[0;4]} = [-2;2]$ $\quad \pm\sqrt{[-4;4]} = [-2;2]$ $\quad \pm\sqrt{[1;4]} = [-2;-1] \cup [1;2]$

Definition (Interval square root)

For all $A = [\underline{A}, \overline{A}] \in \mathbb{I}$ we define $[v_1, v_2] = A \cap [0, +\infty]$ and

$$\pm\sqrt{A} = \begin{cases} \emptyset & \textit{if } v_2 < v_1 \\ \left[-\sqrt{v_2}, +\sqrt{v_2}\right] & \textit{if } v_1 = 0 \ (\textit{with } \sqrt{+\infty} = +\infty) \\ \left[-\sqrt{v_2}, -\sqrt{v_1}\right] \cup \left[\sqrt{v_1}, \sqrt{v_2}\right] & \textit{else.} \end{cases}$$

This can be generalised to arbitrary powers A^k and roots $\sqrt[k]{A}$.

Interval arithmetic: Division

Example (Interval division for $0 \notin B$)

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

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

We define $A \div B = A \cdot \frac{1}{B} = A \cdot [\frac{1}{\overline{B}}; \frac{1}{\underline{B}}]$ for all $A = [\underline{A}, \overline{A}] \in \mathbb{I}$ and $B = [\underline{B}, \overline{B}] \in \mathbb{I}$ with $0 \notin B$.

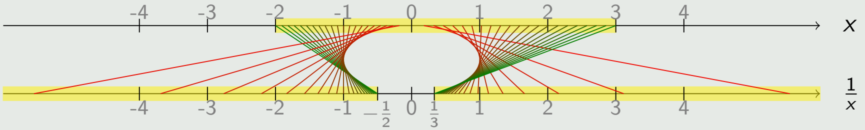
Interval arithmetic: Division

Problem: B may contain 0, but division by 0 is not defined

Example (Interval division for $0 \in B$)

If $0 \in B$ then the previous definition does not work correctly:
$\frac{1}{[-2;3]} = [\frac{1}{3}; -\frac{1}{2}] \rightarrow$ invalid bounds

How should $\frac{1}{[-2;3]}$ be defined?



We observe: $\frac{1}{[-2;3]} = [-\infty; -\frac{1}{2}] \cup [\frac{1}{3}; +\infty]$!
Note: Resulting interval may contain a gap!

Interval arithmetic: Division

Definition (Interval division $A \div B$ for $0 \in B$)

The following table defines the result of $A \div B$ for $0 \in B$; rows define case distinctions on A, columns on B:

$A \div B$	$B = [0, 0]$	$\underline{B} < \overline{B} = 0$	$\underline{B} < 0 < \overline{B}$	$0 = \underline{B} < \overline{B}$
$0 \in A$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$	$(-\infty, +\infty)$
$\overline{A} < 0$	$\emptyset$	$[\overline{A}/\underline{B}, +\infty]$	$[-\infty, \overline{A}/\overline{B}] \cup [\overline{A}/\underline{B}, +\infty]$	$[-\infty, \overline{A}/\overline{B}]$
$0 < \underline{A}$	$\emptyset$	$[-\infty, \underline{A}/\underline{B}]$	$[-\infty, \underline{A}/\underline{B}] \cup [\underline{A}/\overline{B}, +\infty]$	$[\underline{A}/\overline{B}, +\infty]$

How to strenthen bounds using interval arithmetic

- Now we can compute with intervals.
- Remember that the input of ICP (as a theory solver in an SMT solver) is a set C of QFNRA constraints in n ordered variables $x_1, \ldots, x_n$ and an initial box $B = A_1 \times \ldots \times A_n$ (interval domains A_i for the variables x_i in the constraints).
- Our goal is to decide whether the initial box B contains a common satisfying solution for the constraints in C.
- Let us first have a look at how we can make the initial box B smaller without loosing any solutions.

  This bound strengthening is done via propagation.
- We learn two different propagation methods.

Propagation I: Preprocessing

- The first propagation method requires that for each $c \in C$ and each variable x in c, we can bring c to an equivalent form $x \sim e$ with $\sim \in \{<, \leq, =, \geq, >\}$, where x does not appear in e.
- This is doable for linear constraints, and also for equations with only multiplication operators if we allow division and root operations in e.
- We need some preprocessing (done for each constraint one time, when ICP receives it) to satisfy this requirement.

Preprocessing: Example

1. $x^2 \cdot y + z = 0 \quad \rightarrow \quad h + z = 0 \wedge h_1 = x^2 \cdot y$
2. Now the constraints satisfy the requirements:

$$
\begin{array}{lll}
h + z = 0 & \rightarrow & h = -z \\
 & \rightarrow & z = -h
\end{array}
\qquad
\begin{array}{lll}
h_1 = x^2 \cdot y & \rightarrow & h_1 = x^2 \cdot y \\
 & \rightarrow & x = \pm\sqrt{h_1 \div y} \\
 & \rightarrow & y = h_1 \div (x^2)
\end{array}
$$

Propagation I: Preprocessing

- Set $C' := C$ and $C := \emptyset$.
- Repeat as long as C' is not empty:
  - Take a constraint $e_1 \sim e_2$, $\sim \in \{<, \leq, =, \geq, >\}$, from C'.
  - Bring $e_1 \sim e_2$ to the normal form $r_1 \cdot m_1 + \ldots + r_k \cdot m_k \sim 0$, where $r_i \in \mathbb{R}$ and m_i are monomials (either 1 or a product of variables) for each $i = 1, \ldots, k$.
  - Replace each non-linear monomial m_i in $r_1 \cdot m_1 + \ldots + r_k \cdot m_k \sim 0$ by a fresh variable h_i and add the result to C.
  - For each newly added variable h_i replacing m_i in the previous step, add an equation $h_i - m_i = 0$ to C, and initialize the bounds of h_i to the interval we get when we substitute the variable bounds in m_i and evaluate the result using interval arithmetic (note: the result will always be a single interval because there is no division or square root in m_i).

Propagation I: Method

- Choose a constraint $c \in C$ and a variable x appearing in c.

  We call such a pair (c, x) a contraction candidate (CC).
- Bring c to a form $x \sim e$, $\sim \in \{<, \leq, =, \geq, >\}$, where e does not contain x. (Note: due to preprocessing, if c is non-linear then it is of the form $h - m = 0$ with h a variable and m a monomial.)
- Replace all variables in e by their current bounds.
- Apply interval arithmetic to evaluate the right-hand-side (e with the variables substituted by their bounds) to a union of intervals.
- Make a case distinction for each interval B in that union.
- For each case, derive from the current bound A for x and the computed bound B for e a new bound on x, depending on the type of $\sim$, as follows:

$$
\begin{array}{lll}
x < e & \textit{if } \underline{A} \geq \overline{B} \textit{ then } \emptyset \textit{ else} & [\underline{A}, \min\{\overline{A}, \overline{B}\}] \\
x \leq e & & [\underline{A}, \min\{\overline{A}, \overline{B}\}] \\
x = e & & [\max\{\underline{A}, \underline{B}\}, \min\{\overline{A}, \overline{B}\}] \\
x \geq e & & [\max\{\underline{A}, \underline{B}\}, \overline{A}] \\
x > e & \textit{if } \overline{A} \leq \underline{B} \textit{ then } \emptyset \textit{ else} & [\max\{\underline{A}, \underline{B}\}, \overline{A}]
\end{array}
$$

Propagation I: Method

Example (Propagation)

$x \in [1;3], y \in [1;2], c_1 : y = x, c_2 : y = x^2$
$c_2, x : x = \pm\sqrt{y} \to x = \pm\sqrt{[1;2]} = [-\sqrt{2}; -1]] \cup [1; \sqrt{2}] \to$
$\qquad x \in [1;3] \cap ([-\sqrt{2}; -1] \cup [1; \sqrt{2}]) = [1; \sqrt{2}]$
$c_1, y : y = x \to y = [1; \sqrt{2}] \to y \in [1;2] \cap [1; \sqrt{2}] = [1; \sqrt{2}]$

Propagation II: Preprocessing

- Now we look at an alternative method for propagation.
- This method is called the interval Newton method.
- Also this second propagation method needs some lightweight preprocessing:
  - Transform each constraint $e_1 \sim e_2$ in C to $e_1 - e_2 \sim 0$.
  - For each inequation $p \sim 0$ with $\sim \in \{<, \leq, \geq, >\}$ in C, replace p by a fresh variable h, add an equation $h - p = 0$ to C, and initialize the bounds of h to the interval we get when we substitute the variable bounds in p and evaluate the result using interval arithmetic (note: the result will always be a single interval because there is no division or square root in p).
- After this preprocessing, the constraint set contains equations $p = 0$ stating that a polynomial equals to zero, and inequations of the form $x \sim 0$ with x a variable and $\sim \in \{<, \leq, \geq, >\}$.
- Assume in the following a constraint $c \in C$ and a variable x in c as a contraction candidate.
- Next we see how we can reduce the domain of x using c via the interval Newton method.

Propagation II: Method

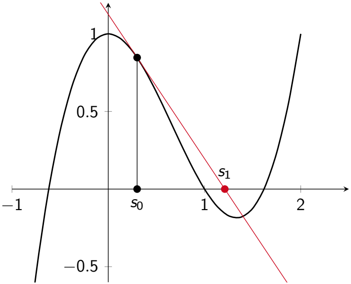
Due to the preprocessing, if the constraint c is an inequation then it has the form $x \sim 0$ (where x is a variable). In this case we propagate similarly as with the first method, assuming that the current interval for x is A:

$$\begin{array}{lll}
x < 0 & \textit{if } \underline{A} \geq 0 \textit{ then } \emptyset \textit{ else} & [\underline{A}, \min\{\overline{A}, 0\}] \\
x \leq 0 & & [\underline{A}, \min\{\overline{A}, 0\}] \\
x \geq 0 & & [\max\{\underline{A}, 0\}, \overline{A}] \\
x > 0 & \textit{if } \overline{A} \leq 0 \textit{ then } \emptyset \textit{ else} & [\max\{\underline{A}, 0\}, \overline{A}]
\end{array}$$

Assume now that the constraint c is $f(x) = 0$, where $f(x) : \mathbb{R} \to \mathbb{R}$ is an univariate polynomial in x, and let $f'(x) : \mathbb{R} \to \mathbb{R}$ be the first derivative of $f(x)$.

Reminder for Newton method for root finding (univariate case): Compute a sequence of real values $s_0, s_1, \ldots$ such that $s_0 \in \mathbb{R}$ is an initial guess, and $s_{i+1} = s_i - \frac{f(s_i)}{f'(s_i)}$ for all $i \geq 0$.

For a "good enough" initial guess s_0, the sequence converges to a zero $r \in \mathbb{R}$ of $f(x)$, i.e., to a value r for which $f(r) = 0$. If it converges then it does so quadratically. Unfortunately, this procedure can be unstable near a horizontal asymptote or a local extremum.



$$
\begin{aligned}
f(x) &= x^3 - 2x^2 + 1 \\
f'(x) &= 3x^2 - 4x \\
s_0 &= 0.3 \\
s_1 &= s_0 - \frac{f(s_0)}{f'(s_0)} \\
&= 0.3 - \frac{f(0.3)}{f'(0.3)} \\
&= 0.3 - \frac{0.847}{-0.93} \\
&\approx 1.2107
\end{aligned}
$$

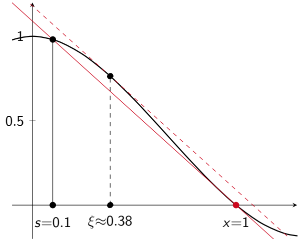
Propagation II: Taylor's Theorem

The interval Newton method is an extension of the Newton method. It takes a function $f : \mathbb{R} \to \mathbb{R}$ which is continuously differentiable on an interval A (polynomials satisfy this condition) and a sample point $s \in A$, and uses information about $f(s)$ and the range of f' on A to contract the set of possible zeros of f within A.

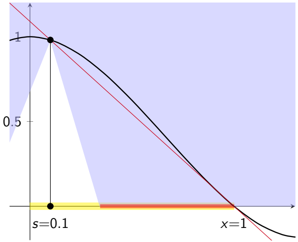
We make use of the first-order version of Taylor's theorem which states that

$$\forall s, x \in A. \exists \xi \in A. f(x) = f(s) + (x - s) f'(\xi).$$

That means, if we take an arbitrary point $s \in A$ then for any $x \in A$ with $f(x) = 0$, the gradient of the line connecting the points $(s, f(s))$ and $(x, 0)$ is in the interval $f'(I)$.



Interval extension of Newton's method:



Function: $f(x) = x^3 - 2x^2 + 1$, $f'(x) = 3x^2 - 4x$

Starting interval: $A = [0;1]$

Sample point: $s = 0.1$

Derivatives in A: $f'(A) = 3 \cdot [0;1]^2 - 4 \cdot [0;1] = [-4;3]$

Possible zeros in A at: $s - \frac{f(s)}{f'(A)} = [-\infty; -0.227] \cup [0.34525; +\infty]$

New interval: $A = [0;1] \cap ([-\infty; -0.227] \cup [0.34525; +\infty]) = [0.34525; 1]$

Propagation II: Componentwise multivariate interval Newton

Reminder: Componentwise Multivariate Newton
Variables $x = (x_1, \ldots, x_n)$, function $f(x)$, sample $s_i = (s_{i,1}, \ldots, s_{i,n})$
$s_{i+1} = N_{cmp}(s, f(x), x_j) = s_i - \frac{f(s_i)}{\frac{\partial f}{\partial x_j}(s_i)}$

Componentwise multivariate interval Newton:
interval $A = A_1 \times \ldots A_n$

$$s_{i+1} = N_{cmp}(A, s_i, \overbrace{f(x), x_j}^{CC}) :=$$
$$s_i - \frac{f(A_1, \ldots, A_{j-1}, s_i, A_{j+1}, \ldots, A_n)}{\frac{\partial f}{\partial x_j}(A_1, \ldots, A_n)}$$

The operator N_{cmp} has two important properties:

- If $f(x^*) = 0$ and $x^* \in A$, then $x^* \in N_{cmp}(A, f(x), x_j)$.
- If $A \cap N_{cmp}(A, f(x), x_j) = \varnothing$ then $f(x) \neq 0$ for all $x \in A$.

$\rightarrow$ Advantage: No diverging behavior like the original Newton method due to interval arithmetic.
$\rightarrow$ We can drop boxes when they contract to empty.

The global ICP algorithm

- Now we know how to reduce the bounds of a variable based on a constraint in which it appears.
- Next we look how to use these reduction methods iteratively in an algorithm, which can be used as a theory solver for QFNRA constraint sets in an SMT solver.

Algorithm

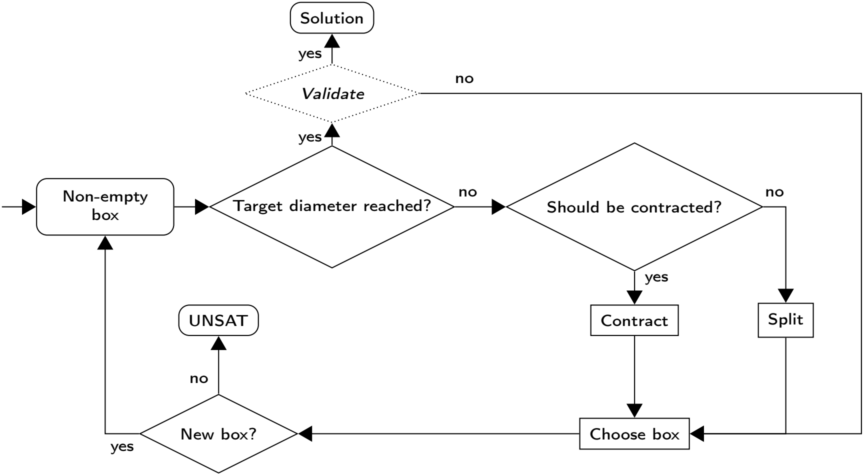
Input: Set of QFNRA constraints, non-empty initial box B_0
Box diameter threshold D, contraction condition for boxes (fix later)

Algorithm

Compute a set of boxes B whose union contains all solutions from B_0 (if any) by iteratively executing one of the following steps:

1. Set $B := \{B_0\}$.
2. If B is empty then return unsatisfiable.
   Otherwise choose a box $B_i \in B$ and remove it from B.
3. If the diameter of B_i is at most D then pass on B_i to a complete procedure for satisfiability check; if B_i contains a solution then return SAT otherwise go to **??**.
4. If the contraction condition for B_i holds then try to reduce this box, add the resulting box(es) to B, and go to **??**. Note: Due to interval division or square root propagation may result in two boxes.
5. Otherwise split the box into two halves, add them to B, and go to **??**.

Algorithm


Solution
yes
Validate
no
yes
Non-empty box
Target diameter reached?
no
Should be contracted?
no
yes
Contract
Split
UNSAT
no
New box?
yes
Choose box

Algorithm

Further algorithmic aspects:

- Heuristics to choose CCs (constraints and variables)
- Assure termination
- ICP does not behave well on linear constraints
- ICP needs to return an explanation for unsatisfiable problems

Heuristics to choose CCs

General approach: Contract via interval constraint propagation.
Problems:

- Contraction gain 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:

- $I_x \stackrel{c_1,x}{\rightarrow} [1; 2]$ $\quad (gain_{rel} : 0.5)$
- $I_y \stackrel{c_1,y}{\rightarrow} [1; 2]$ $\quad (gain_{rel} : 0)$
- $I_x \stackrel{c_2,x}{\rightarrow} [1; \sqrt{2}]$ $\quad (gain_{rel} : 0.793)$
- $I_y \stackrel{c_2,y}{\rightarrow} [1; 2]$ $\quad (gain_{rel} : 0)$

Relative contraction:

$$gain_{rel} = \frac{D_{old} - D_{new}}{D_{old}} = 1 - \frac{D_{new}}{D_{old}}$$

$\rightarrow$ Contraction gain varies.

Heuristics to choose CCs

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.

Assure termination

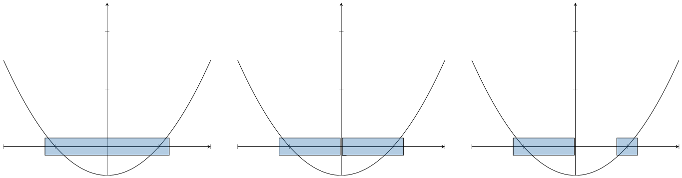
Example (Propagation)

$x \in [1;3], y \in [1;2], c_1 : y = x, c_2 : y = x^2$
$c_2, x : x = \pm\sqrt{y} \to x = \pm\sqrt{[1;2]} = [-\sqrt{2}; -1]] \cup [1; \sqrt{2}] \to$
$\quad x \in [1;3] \cap ([-\sqrt{2}; -1] \cup [1; \sqrt{2}]) = [1; \sqrt{2}]$
$c_1, y : y = x \to y = [1; \sqrt{2}] \to y \in [1;2] \cap [1; \sqrt{2}] = [1; \sqrt{2}]$

Contraction sequence:
$x : [1;3] \stackrel{c_2, x}{\to} [1; \sqrt{2}] \stackrel{c_2, x}{\to} [1; \sqrt[4]{2}] \stackrel{c_2, x}{\to} [1; \sqrt[8]{2}] \stackrel{c_2, x}{\to} \ldots \leadsto [1;1]$
$y : [1;2] \stackrel{c_1, y}{\to} [1; \sqrt{2}] \stackrel{c_1, y}{\to} [1; \sqrt[4]{2}] \stackrel{c_1, y}{\to} [1; \sqrt[8]{2}] \stackrel{c_1, y}{\to} \ldots \leadsto [1;1]$
$\to$ Propagation might not terminate!

Assure termination

When the weight of all CCs is below the threshold we do not make progress $\rightarrow$ split the box.



Handling linear constraints

ICP is not well-suited for linear problems (slow convergence).

Make use of linear solvers (e.g. simplex) for linear constraints:

- Pre-process to separate linear and nonlinear constraints
- Use nonlinear constraints for contraction
- Validate resulting boxes against linear feasible region (by checking the satisfiability of the linear constraints with the constraints defining the box)
- In case box is linear infeasible: Add violated linear constraint for contraction

Explanations

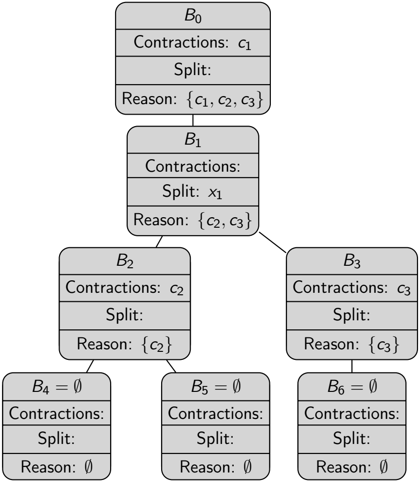
To keep track of current status we utilize a tree-structure, which holds solver states:

- Search box
- Applied contractions or
- Dimension of applied splitting

We can use the tree to collect infeasible subsets:

- Infeasible box $\rightarrow$ propagate reasons to parent

$\rightarrow$ We generate a set of constraints which implies infeasibility.



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

If you like to see a video about ICP:
http://www-sop.inria.fr/coprin/logiciels/ALIAS/Movie/movie_undergraduate.mpg