Satisfiability Checking

19 Interval constraint propagation II

Prof. Dr. Erika Ábrahám

RWTH Aachen University
Informatik 2
LuFG Theory of Hybrid Systems

WS 22/23

Previous lecture:

- Interval arithmetic
- Contraction I

1 Contraction II

2 The global ICP algorithm

Previous lecture:

- Interval arithmetic
- Contraction I

1 Contraction II

2 The global ICP algorithm

Contraction 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).

即: h=p=0或h=p~0

- 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 $\sim \in \{<, \leq, \geq, >\}$.

此处的x为h, h为fresh variable也为变量

- Assume in the following a constraint $c \in C$ and a variable x in c as a contraction candidate (c, x).

h此时有bound, 现需要对h或其他变量x的范围进行缩小.
如果 h/x=p~0, 则用方法一, 将h上界/下界和0比较缩小h范围
如果h=P=0, 则对h中包含的变量x范围进行缩小

Contraction II: Method

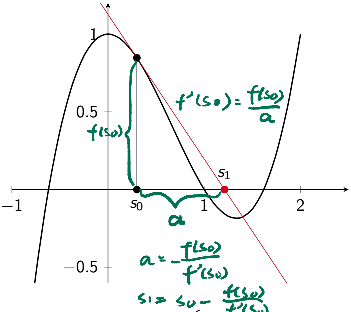
如果是x~ 0, 使用第一种方法

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:

$$
\begin{array}{lll}
x < 0 & \textit{if } \underline{A} \geq 0 \textit{ then } [1;0] \textit{ else} & [\underline{A}; \min\{\overline{A}, 0\}] \\
x \leq 0 & & [\underline{A}; \min\{\overline{A}, 0\}] \\
x \geq 0 & & [\max\{\underline{A}, 0\}; \overline{A}] \\
x > 0 & \textit{if } \overline{A} \leq 0 \textit{ then } [1;0] \textit{ else} & [\max\{\underline{A}, 0\}; \overline{A}]
\end{array}
$$

Contraction II: Method

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 $f(x)$.
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 root $r \in \mathbb{R}$ 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.

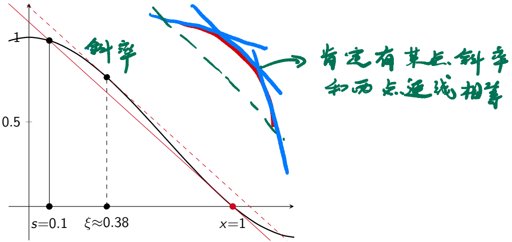


$$
\begin{aligned}
f(x) &= x^3 - 2x^2 + 1 \\
f'(x) &= 3x^2 - 4x \\
s_0 &= 0.3 \\
s_1 &= s_0 - \tfrac{f(s_0)}{f'(s_0)} \\
&= 0.3 - \tfrac{f(0.3)}{f'(0.3)} \\
&= 0.3 - \tfrac{0.847}{-0.93} \\
&\approx 1.2107
\end{aligned}
$$

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 roots of f within A.
We make use of the first-order version of Taylor's theorem which states that

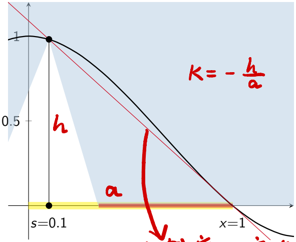
$$\forall s, x \in A.\ \exists \xi \in A.\ f(x) = f(s) + (x - s)f'(\xi).$$

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 $f'(A)$.



Contraction II: Interval Newton method

Interval extension of Newton's method: Sample 在 A 内取值



$a = -\frac{h}{K}$

$$A := A \cap \left(s - \frac{f(s)}{f'(A)}\right)$$

斜率 K

此斜率必定在f'(A)范围内

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

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

Contraction II: Componentwise multivariate interval Newton

The method can be extended to multivariate problems, but we do not discuss the multivariate case in this lecture.

Previous lecture:

- Interval arithmetic
- Contraction I

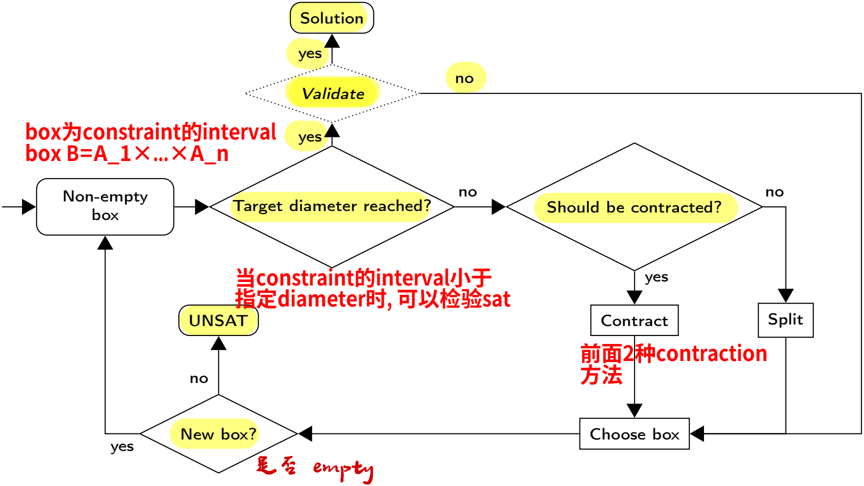
1 Contraction II

2 The global ICP algorithm

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


Solution
yes
Validate
no
yes
box为constraint的interval
box B=A_1×…×A_n
Non-empty box
Target diameter reached?
no
Should be contracted?
no
当constraint的interval小于
指定diameter时, 可以检验sat
yes
UNSAT
Contract
Split
前面2种contraction
方法
no
New box?
Choose box
yes
是否 empty

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 $\mathcal{B}$ whose union contains all solutions from B_0 (if any) by executing the following algorithm:

1. Set $\mathcal{B} := \{B_0\}$.
2. If $\mathcal{B}$ is empty then return unsatisfiable.
   Otherwise choose a box $B_i \in \mathcal{B}$ and remove it from $\mathcal{B}$.
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 2.
4. If the contraction condition for B_i holds then try to reduce this box, add the resulting box(es) to $\mathcal{B}$, and go to 2. Note: Due to interval division or square root propagation this step may result in adding two boxes.
5. Otherwise split the box into two halves, add them to $\mathcal{B}$, and go to 2.

Algorithmic aspects

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

Algorithmic aspects

- Heuristics to choose CCs (constraints and variables)
- Assure termination
- ICP does not behave well on linear constraints
- ICP needs to work incrementally
- 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

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:

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

Relative contraction:

$$gain_{rel} = \frac{D(old) - D(new)}{D(old)} = 1 - \frac{D(new)}{D(old)}$$

gain=缩小的D/原来的D

$\rightarrow$ 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 $W_k^{(ij)} \in [0;1]$.
  constrain i var j
- Afterwards the weight is updated (according to the relative contraction $r_{k+1}^{(ij)} \in [0;1]$).

Weight updating:

gain

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

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.

Algorithmic aspects

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

Assure termination

Example (Propagation)

$x \in [1;3]$, $y \in [1;2]$, $c_1 : y = x$, $c_2 : y = x^2$

(c_2, x): $x = \pm\sqrt{y} \rightarrow x = \pm\sqrt{[1;2]} = [-\sqrt{2}; -1] \cup [1; \sqrt{2}] \rightarrow$

$\quad x \in [1;3] \cap ([-\sqrt{2}; -1] \cup [1; \sqrt{2}]) = [1; \sqrt{2}]$

(c_1, y): $y = x \rightarrow y = [1; \sqrt{2}] \rightarrow y \in [1;2] \cap [1; \sqrt{2}] = [1; \sqrt{2}]$

Contraction sequence:

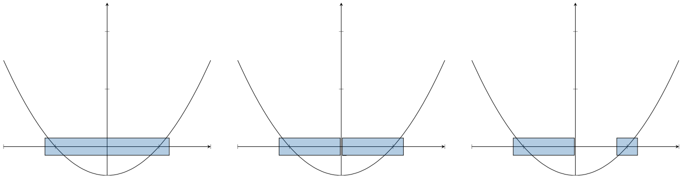
$x: \ [1;3] \stackrel{c_2,x}{\rightarrow} [1; \sqrt{2}] \stackrel{c_2,x}{\rightarrow} [1; \sqrt[4]{2}] \stackrel{c_2,x}{\rightarrow} [1; \sqrt[8]{2}] \stackrel{c_2,x}{\rightarrow} \ldots \rightsquigarrow [1;1]$

$y: \ [1;2] \stackrel{c_1,y}{\rightarrow} [1; \sqrt{2}] \stackrel{c_1,y}{\rightarrow} [1; \sqrt[4]{2}] \stackrel{c_1,y}{\rightarrow} [1; \sqrt[8]{2}] \stackrel{c_1,y}{\rightarrow} \ldots \rightsquigarrow [1;1]$

$\rightarrow$ 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.



Algorithmic aspects

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

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

Algorithmic aspects

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

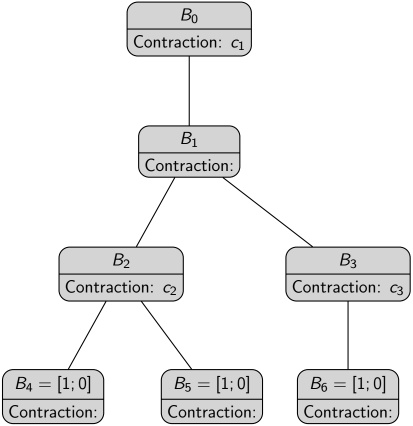
Incrementality and explanations

We store the search history in a tree-structure. Each node stores information about one loop iteration:

- the box chosen and
- the constraint used for contraction if any.

Incrementality: Extend the tree.

Explanation: collect all constraints mentioned in the tree.



Learning target

- How are intervals defined?
- How are set operations on intervals defined?
- How are arithmetic operations on intervals defined?

- Contraction I: How can we contract the domain of a variable x for a constraint c if we can x to one side of the constraint?
- How can we contract domains otherwise using the interval Newton method?

- How can we use interval constraint propagation to decide the satisfiability of a set of real-arithmetic constraints (in an incomplete manner)?