NLVerify

Non Linear Verification - a COQ-Tactic

Tillmann Weisser

with Victor Magron and Benjamin Werner

MAC, LAAS, CNRS, Toulouse

February 1, 2016

Disclaimer

For the moment, `NLVerify` is only implemented for *polynomial* goals.

Introduction

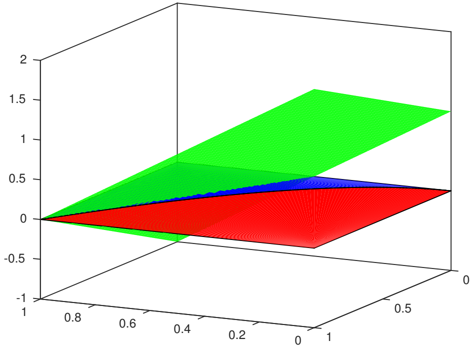
Demo

Implementation

Benchmarks

NLVerify

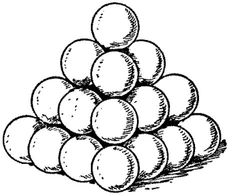
$$B := [0,1]^2, \quad K := B \cap \{(x,y) \mid y \geq x^2\}, \quad f := Y - 2X + 1$$



Prove "f *non negative* on K" - i.e. $\forall (x,y) \in K : f(x,y) \geq 0$

Applications

Formal proofs (Kepler Conjecture)



System control

Software verification

Existing Methods

- Taylor/Interval approach
- Bernstein polynomials
- Quantifier elimination
- Naive interval enclosure
- Sums of Square certificates

Naive Interval Enclosure

$$B := [0,1]^2, \quad K := B \cap \{(x,y) \mid y \geq x^2\}, \quad f := Y - 2X + 1$$

Replace everything by corresponding intervals and do the interval arithmetics:

$$\boxed{Y - 2X + 1}_B = [0,1] - [2,2][0,1] + [1,1] = [-1,2]$$

$$\Rightarrow Y - 2X + 1 \geq -1 \text{ on } B$$

Does not use the hypothesis $y \geq x^2$.

Sums of squares certificates

$$B := [0,1]^2, \quad K := B \cap \{(x,y) \mid y \geq x^2\}, \quad f := Y - 2X + 1$$

Compute a representation of the objective function that is obviously non negative on K:

$$Y - 2X + 1 = (X-1)^2 + (Y - X^2)$$

$$\Rightarrow Y - 2X + 1 \geq 0 \text{ on } K$$

SOS-representation can be found by solving an SDP.

Problems with automatically generated certificates

- Usually only $f \approx \text{cert}$ due to finite precision of the solver.
- Verifying $f = \text{cert}$ is time consuming in COQ.

`NLVerify` addresses both problems by considering a naive interval enclosure of $f - \text{cert}$ using verified floating points (`Flocq`) and interval arithmetics ($\texttt{coq} - \texttt{interval}$).

Example: Compose SOS and interval enclosure

$$B := [0,1]^2, \quad K := B \cap \{(x,y) \mid y \geq x^2\}, \quad f := Y - 2X + 1$$

Consider the approximate numerical certificate

$$\begin{aligned} \text{cert} &= 2.00014(0.707263X + 0.000078Y - 0.70695)^2 \\ &+ 0.000332(-0.408035X + 0.816664Y - 0.408126)^2 \\ &+ 0.000284Y + 0.000116(1 - Y) + 1.00034(Y - X^2) \end{aligned}$$

Then $\boxed{f - \text{cert}}_B = \boxed{2.32209 \times 10^{-4}X^2 - 5.81334 \times 10^{-7}XY}$ $\boxed{-2.97356 \times 10^{-5}X + 2.21436 \times 10^{-4}Y^2 + 6.21035 \times 10^{-5}Y}$ $\boxed{-2.01126 \times 10^{-4}}_B \subseteq \left[-2.3144 \times 10^{-4}, 5.1575 \times 10^{-4}\right]$

$$\Rightarrow f \geq -2.3144 \times 10^{-4} \text{ on } K$$

`NLVerify` – an incomplete automatic tactic

Procedure: (call `NLVerify`)

- if tactic not applicable, return failed
  else, request certificate from oracle
- if oracle fails, return failed
  else, compute *checker function* with goal and certificate
- if checker function returns `false`, return failed
  else, apply `Lemma checker_ok` which solves the goal.

Demo

Some details on the implementation

This talk:

$\rightarrow$ Definition and correctness of an interval enclosure

$$\begin{tikzcd} p \arrow[r, "\text{eval } x"] \arrow[dr, "\text{encl } B"'] & p(x) \arrow[d, "\in"] \\ & \boxed{p}_B \end{tikzcd}$$

for all $x \in B$.

Criteria to be considered:

- Time performance
- Tightness

Types

- `NLVerify` solves goals about COQ-reals `R`.
- Data is given by COQ-rationals `Q`.
- Intervals `I` are represented by `coq` − `interval` with floating point bounds `Flocq` (radix 2, precision ρ)
- Polynomials are represented by

```
Inductive PQ :=
| PEc   : Q → PQ
| PEx   : positive → PQ
| PEadd: PQ -> PQ → PQ
| PEsub: PQ -> PQ → PQ
| ... .
```

Theorem

```
Lemma (p:PQ) (x: seq R) (B: seq Q*Q):
x ∈ B → (eval x p) ∈ (encl B p).
```

$$\begin{tikzcd}
\mathtt{PQ} \arrow[r, "\mathtt{eval\,x}"] \arrow[dr, "\mathtt{encl\,B}"'] & \mathtt{R} \arrow[d, "\in"] \\
 & \mathtt{I}
\end{tikzcd}$$

Definition of an interval enclosure

We use two different kinds of enclosures:

- The *coefficient enclosure* $[p]$ (`cencl`)
  - maps a polynomial to a set of polynomials
  - $\left[\frac{1}{3}X+2\right] = [0.33, 0.34]X + [2,2] := \{aX + b \mid a \in [0.33, 0.34], b \in [2,2]\}$
- The *variable enclosure* $\tilde{p}\Big|_B$ (`vencl`)
  - maps a set of polynomials to an interval
  - $[0.33, 0.34]X + [2,2]\big|_{[0,1]} = [0.33, 0.34][0,1] + [2,2] = [2, 2.34]$

The coefficient enclosure

```
Inductive PI : Type:=
|IPEc   : I → PI
|IPEx   : positive → PI
|IPEadd: PI -> PI → PI
|... .

Fixpoint cencl p : PI:= match p with
|PEc q => IPEc (Q2I q)
|PEX j => IPEX j
|PEadd P Q => IPEadd (cencl P) (cencl Q)
|... end.
```

Preliminary Consideration for the variable enclosure

Let $B = [-1, 1] \times [0, 1] \times [0, 1]$.

$$\begin{array}{rcl} X(Y-Z) & = & XY - XZ \\ [X(Y-Z)]\Big|_B & = & [-1,1][-1,1] = [-1,1] \\ [XY - XZ]\Big|_B & = & [-1,1] - [-1,1] = [-2,2] \end{array}$$

The variable enclosure

`NLVerify` uses a Horner type representation to build the variable enclosure:

```
Inductive PolI : Type :=
|IPc   : I -> PolI
|IPinj: positive -> PolI -> PolI
|IPX   : PolI -> positive -> PolI -> PolI.

Fixpoint vencl B p : I := match p with
|IPc c => c
|IPinj j Q => vencl (jump j B) Q
|IPX P i Q => (vencl B P) * ((hd B) ^ i)
   + vencl (tail B) Q
end.
```

The interval enclosure in `NLVerify`

$$\begin{tikzcd}
\mathtt{PQ} \arrow[rr, "\mathtt{eval\,x}"] \arrow[d, "\mathtt{cencl}"] & & \mathtt{R} \arrow[d, "\in"] \\
\mathtt{PI} \arrow[r, "\mathtt{horner}"] & \mathtt{PolI} \arrow[r, "\mathtt{vencl\,B}"] & \mathtt{I}
\end{tikzcd}$$

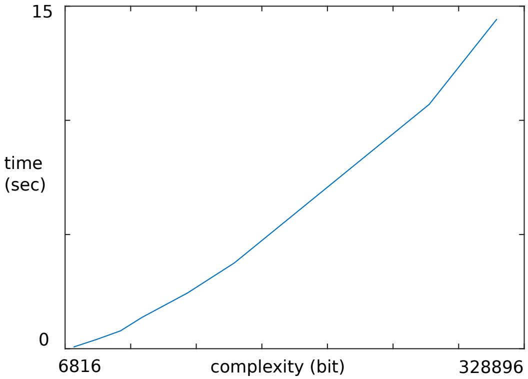
The interval enclosure in `NLVerify`

$$\begin{tikzcd}
\mathtt{PQ} \arrow[rr, "\mathtt{eval}\,\mathtt{x}"] \arrow[dr, "\mathtt{horner}'"] \arrow[dd, "\mathtt{cencl}"] & & \mathtt{R} \arrow[d, "="] \\
& \mathtt{PolQ} \arrow[r, "\mathtt{eval}'\,\mathtt{x}"] \arrow[d, "\mathtt{cencl}'"] & \mathtt{R} \arrow[d, "\in"] \\
\mathtt{PI} \arrow[r, "\mathtt{horner}"] & \mathtt{PolI} \arrow[r, "\mathtt{vencl}\,\mathtt{B}"] & \mathtt{I}
\end{tikzcd}$$

NLVerify

$$\frac{\begin{array}{l}\text{parse goal}\\ \text{ask for certificate}\\ \texttt{checker\_pop(goal,cert)= true}\end{array}}{\texttt{conclude goal = True}}$$

Total time Checker function



Examples range from (variables, degree) $= (6, 2)$ to $(3, 8)$.

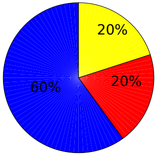
Checker Function

The function `checker_pop` computes:

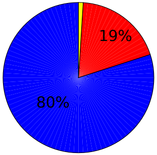
- rewrite goal and certificate $\rightarrow$ `PQ`
- `cencl`
- `horner`
- `vencl`
- check whether the resulting interval is non negative

If `checker_pop` returns `true`, `NLVerify` concludes by `Lemma checker_ok` that the assertion is `True`.

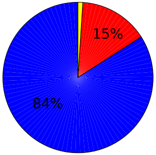
Time consumption Checker function



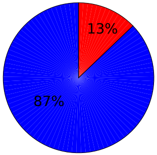
(a) ex621
6816 bit



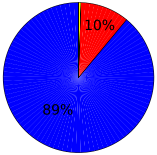
(b) ex261
23776 bit



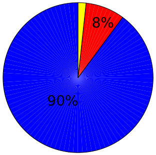
(c) ex441
42176 bit



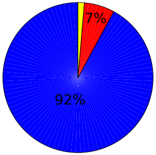
(d) ex264
58560 bit



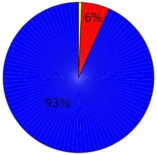
(e) ex361
93312 bit



(f) ex622
129024 bit



(g) ex443
277568 bit



(h) ex364
328896 bit

Questions

The choice of using Horner representation for the variable enclosure is arbitrary.

- Is there a representation, which results generically in a tighter enclosure?
- Is there a representation, which can be computed faster?

We are currently working on

- Generalization to semi algebraic functions (using generalized polynomial optimization)
- Generalization to univariate transcendental functions (using polynomial approximations verifyed by $\mathtt{coq} - \mathtt{interval}$)

Thanks for your attention.