$\mathbb{R}$eal Number Proving in PVS

César A. Muñoz

NASA Langley Research Center
Cesar.A.Munoz@nasa.gov

PVS Tutorial 2017

Outline

Why Real Number Proving in PVS ?

- Real numbers appear in *real* applications, i.e., cyber-physical systems.
- Conceptually, it is easier to reason on a continuous framework than on a discrete one.
- Availability of many classical results in calculus, trigonometry, and continuous mathematics.

Computer Algebra Systems (CAS) and Theorem Provers

- Mathematica, Maple, Matlab, etc. provide very powerful symbolic and numerical engines.
- These systems **do not** claim to be *logically sound*. Singularities and exceptions are well-known problems of CAS.
- CAS provide programming languages (as opposed to *specification languages*.)
- Real analysis is not a traditional strength of theorem provers.
  - CAS can be used to perform mechanical simplifications and find potential solutions.
  - A theorem prover can be used to verify the correctness of a particular solution.

Real Numbers in PVS

- Reals are defined as an uninterpreted subtype of `number` in the prelude library:

  ```
  real: TYPE+ FROM number
  ```

- All numeric constants are `real`:
  - naturals: `0,1,...`
  - integers: `...,-1,0,1,...`
  - rationals: `...,-1/10,...,3/2,...`

- Decimal notation is supported: The decimal number `3.141516` is syntactic sugar for the rational number `31416/10000`.

PVS's `real` numbers are $\mathbb{R}$eal

- All the standard properties: unbounded, connected, infinite, $\mathbb{N} \subseteq \mathbb{Z} \subseteq \mathbb{Q} \subseteq \mathbb{R}$, ...
- $\mathbb{R}$eal arithmetic: `1/3 + 1/3 + 1/3 = 1`.
- The type `real` is unbounded:

```
googol     : real = 10^100
googolplex : real = 10^googol

googol_prop : LEMMA
  googolplex > googol * googol
```

- ...but *machine physical limitations do apply*, e.g., don't try to prove googol_prop with `(grind)`.

Rational Arithmetic is Built-in

```
  |-------
{1} -(0.78 * 1.05504 * (0.92 - 0.78) * s) -
         0.78 * 1.08016 * (0.9 - 0.78) * s
         - 1.256 * (0.9 - 0.78) * s * u
         - 0.92944 * (0.92 - 0.78) * s * u
       + ...
        + 1.05504 * (0.92 - 0.78) * s * u
         + 1.08016 * (0.9 - 0.78) * s * u >= 0

Rule? (assert)
 |-------
{1} 0.0052256+-(0.115210368*s)+0.00844032*u+0.154213*s
    - 0.00568*(s*u) >= 0
```

Rational Arithmetic is Built-in

```
  |-------
{1} -(0.78 * 1.05504 * (0.92 - 0.78) * s) -
         0.78 * 1.08016 * (0.9 - 0.78) * s
         - 1.256 * (0.9 - 0.78) * s * u
         - 0.92944 * (0.92 - 0.78) * s * u
       + ...
        + 1.05504 * (0.92 - 0.78) * s * u
         + 1.08016 * (0.9 - 0.78) * s * u >= 0

Rule? (assert)
 |-------
{1} 0.0052256+-(0.115210368*s)+0.00844032*u+0.154213*s
    - 0.00568*(s*u) >= 0
```

Subtypes of `real`

```
nzreal  : TYPE+ = {r:real| r /= 0} % Nonzero reals
nnreal  : TYPE+ = {r:real| r >= 0} % Nonnegative reals
npreal  : TYPE+ = {r:real| r <= 0} % Nonpositive reals
negreal : TYPE+ = {r:real| r < 0} % Negative reals
posreal : TYPE+ = {r:real| r > 0} % Positive reals

rat     : TYPE+ FROM real
int     : TYPE+ FROM rat
nat     : TYPE+ FROM int
```

The uninterpreted type `number` is the only `real`'s supertype predefined in PVS: no complex numbers, no hyper-reals, no $\mathbb{R}^{\infty}$*, ...*

Real Numbers Properties

Real numbers in PVS are axiomatically defined in the prelude:

- Theory `real_axioms`:
  Commutativity, associativity, identity, etc. These properties are known to the decision procedures, so they rarely need to be used in a proof.

- Theory `real_props`:
  Order and cancellation laws. These lemmas are not used automatically by the standard decision procedures.

Theory real_props

```
real_props: THEORY
 BEGIN
  both_sides_plus_le1: LEMMA x + z <= y + z IFF x <= y
  both_sides_plus_le2: LEMMA z + x <= z + y IFF x <= y
  both_sides_minus_le1: LEMMA x - z <= y - z IFF x <= y
  both_sides_minus_le2: LEMMA z - x <= z - y IFF y <= x
  both_sides_div_pos_le1: LEMMA x/pz <= y/pz IFF x <= y
  both_sides_div_neg_le1: LEMMA x/nz <= y/nz IFF y <= x
  ...
  abs_mult: LEMMA abs(x * y) = abs(x) * abs(y)
  abs_div: LEMMA abs(x / n0y) = abs(x) / abs(n0y)
  abs_abs: LEMMA abs(abs(x)) = abs(x)
  abs_square: LEMMA abs(x * x) = x * x
  abs_limits: LEMMA -(abs(x) + abs(y)) <= x + y AND
                    x + y <= abs(x) + abs(y)
END real_props
```

Predefined Operations

```
+, -, *: [real, real -> real]
/: [real, nzreal -> real]
-: [real -> real]

sgn(x:real)  : int = IF x >= 0 THEN 1 ELSE -1 ENDIF
abs(x:real)  : {nny: nnreal | nny >= x} = ...
max(x,y:real): {z: real | z >= x AND z >= y} = ...
min(x,y:real): {z: real | z <= x AND z <= y} = ...
^(x: real,i:{i:int | x /= 0 OR i >= 0}): real = ...
```

...and what about $\sqrt{}, \int, \log, \exp, \sin, \cos, \tan, \pi, \lim, \ldots$ *?*

NASA PVS Libraries

http://github.com/nasa/pvslib

- `reals`: Square, square root, quadratic formula, polynomials.
- `analysis`: Real analysis, limits, continuity, derivatives, integrals.
- `vectors` and `vect_analysis`: Vector calculus and analysis.
- `series`: Power series, Taylor's theorem.
- `trig`: Trigonometric functions.[1]
- `lnexp_fnd`: Logarithm, exponential, and hyperbolic functions.

[1] `trig` has replaced the now deprecated `trig_fnd`.

Beyond Real Numbers

- `complex` and `complex_alt`: Complex numbers.
- `float`: Floating point numbers.
- `interval_arith`: Interval arithmetic.
- `affine_arith`: Affine arithmetic.
- `exact_real_arith`: Exact real arithmetic.
- ...

Basic Real Number Proving

PVS offers some proof commands for simple algebraic manipulations:

```
one_fourth :

  |-------
{1}   x - x * x <= 1

Rule? (both-sides "-" "1/4")
one_fourth :

  |-------
{1}   x - x * x - 1 / 4 <= 1 - 1 / 4
```

Note: Use both-sides only to add/subtract expressions.

Basic Real Number Proving

PVS offers some proof commands for simple algebraic manipulations:

```
one_fourth :

  |-------
{1}   x - x * x <= 1

Rule? (both-sides "-" "1/4")
one_fourth :

  |-------
{1}   x - x * x - 1 / 4 <= 1 - 1 / 4
```

Note: Use `both-sides` only to add/subtract expressions.

Use case to Prove What You Need

```
  |-------
{1}   x - x * x - 1 / 4 <= 1 - 1 / 4

Rule? (case "x - x * x - 1 / 4 <= 0")
this yields  2 subgoals:
one_fourth.1 :
{-1}  x - x * x - 1 / 4 <= 0
  |-------
[1]   x - x * x - 1 / 4 <= 1 - 1 / 4

Rule? (assert)
This completes the proof of one_fourth.1.
```

Use case to Prove What You Need

```
  |-------
{1}   x - x * x - 1 / 4 <= 1 - 1 / 4

Rule? (case "x - x * x - 1 / 4 <= 0")
this yields  2 subgoals:
one_fourth.1 :
{-1}  x - x * x - 1 / 4 <= 0
  |-------
[1]   x - x * x - 1 / 4 <= 1 - 1 / 4

Rule? (assert)
This completes the proof of one_fourth.1.
```

Use case to Prove What You Need

```
  |-------
{1}   x - x * x - 1 / 4 <= 1 - 1 / 4

Rule? (case "x - x * x - 1 / 4 <= 0")
this yields  2 subgoals:
one_fourth.1 :
{-1}  x - x * x - 1 / 4 <= 0
  |-------
[1]   x - x * x - 1 / 4 <= 1 - 1 / 4

Rule? (assert)
This completes the proof of one_fourth.1.
```

Use hide to Focus on Relevant Formulas

```
one_fourth.2 :
  |-------
{1}   x - x * x - 1 / 4 <= 0
[2]   x - x * x - 1 / 4 <= 1 - 1 / 4

Rule? (hide 2)
one_fourth.2 :
  |-------
[1]   x - x * x - 1 / 4 <= 0
```

Use hide to Focus on Relevant Formulas

```
one_fourth.2 :
  |-------
{1}   x - x * x - 1 / 4 <= 0
[2]   x - x * x - 1 / 4 <= 1 - 1 / 4

Rule? (hide 2)
one_fourth.2 :
  |-------
[1]   x - x * x - 1 / 4 <= 0
```

Arrange Expressions With `case-replace`

```
one_fourth.2 :
  |-------
[1]   x - x * x - 1 / 4 <= 0
Rule? (case-replace
       "x - x * x - 1 / 4 = -(x-1/2)*(x-1/2)"
       :hide? t)
this yields  2 subgoals:
one_fourth.2.1 :
  |-------
{1}   -(x - 1 / 2) * (x - 1 / 2) <= 0
```

Arrange Expressions With case-replace

```
one_fourth.2 :
  |-------
[1]   x - x * x - 1 / 4 <= 0
Rule? (case-replace
       "x - x * x - 1 / 4 = -(x-1/2)*(x-1/2)"
       :hide? t)
this yields  2 subgoals:
one_fourth.2.1 :
  |-------
{1}   -(x - 1 / 2) * (x - 1 / 2) <= 0
```

Introduce New Names With `name-replace`

```
one_fourth.2.1 :
  |-------
{1}   -(x - 1 / 2) * (x - 1 / 2) <= 0

Rule? (name-replace "X" "(x-1/2)")
one_fourth.2.1 :


  |-------
{1}   -X * X <= 0


Rule? (assert)
This completes the proof of one_fourth.2.1.
```

Introduce New Names With name-replace

```
one_fourth.2.1 :
  |-------
{1}   -(x - 1 / 2) * (x - 1 / 2) <= 0

Rule? (name-replace "X" "(x-1/2)")
one_fourth.2.1 :

  |-------
{1}   -X * X <= 0

Rule? (assert)
This completes the proof of one_fourth.2.1.
```

Introduce New Names With name-replace

```
one_fourth.2.1 :
  |-------
{1}   -(x - 1 / 2) * (x - 1 / 2) <= 0

Rule? (name-replace "X" "(x-1/2)")
one_fourth.2.1 :

  |-------
{1}   -X * X <= 0

Rule? (assert)
This completes the proof of one_fourth.2.1.
```

Don't Reinvent the Wheel

Look into the NASA PVS libraries first!

Theory `reals@quadratic`:

```
quadratic_le_0 : LEMMA
  a*sq(x) + b*x + c <= 0 IFF
  ((discr(a,b,c) >= 0 AND
    ((a > 0 AND x2(a,b,c) <= x AND x <= x1(a,b,c)) OR
     (a < 0 AND (x <= x1(a,b,c) OR x2(a,b,c) <= x)))) OR
   (discr(a,b,c) < 0 AND c <= 0))
```

A Simpler Proof

```
  |-------
{1}   x * (1 - x) <= 1

Rule? (lemma "quadratic_le_0"
                ("a" "-1" "b" "1" "c" "-1" "x" "x"))
      (grind)
Trying repeated skolemization, instantiation, and
if-lifting,
Q.E.D.
```

An Even Simpler Proof

```
  |-------
{1}   x * (1 - x) <= 1

Rule? (sturm)
Q.E.D.
```

Manip

- Manip is a PVS package for algebraic manipulations of real-valued expressions.
- `http://shemesh.larc.nasa.gov/people/bld/manip.html`.
- The package consists of:
  - Strategies.
  - Extended notations for formulas and expressions.
  - Emacs extensions.
  - Support functions for strategy developers.

Manip Strategies: Basic Manipulations

Strategy	Description
(`swap-rel` *fnums*)	Swap sides and reverse relations
(`swap!` *exprloc*)	$x \circ y \Rightarrow y \circ x$
(`group!` *exprloc* `l|r`)	$(x \circ y) \circ z \Rightarrow x \circ (y \circ z)$
(`flip-ineq` *fnums*)	Negate and move inequalities
(`split-ineq` *fnum*)	Split $\leq$ ($\geq$) into $<$ ($>$) and $=$

Extended Formula Notation

- Standard
  - ∗: All formulas.
  - -: All formulas in the antecedent.
  - +: All formulas in the consequent.
- Extended (Manip strategies only)
  - (^ $n_1 \ldots n_k$): All formulas but $n_1, \ldots, n_k$
  - (-^ $n_1 \ldots n_k$): All antecedent formulas but $n_1, \ldots, n_k$
  - (+^ $n_1 \ldots n_k$): All consequent formulas but $n_1, \ldots, n_k$

(Basic) Extended Expression Notation

- Term indexes:
  - `l`,`r`: Left- or right-hand side of a formula.
  - *n*: *n*-th term from left to right in a formula.
  - $-n$: *n*-th term from right to left in a formula.
  - `*`: All terms in a formula.
  - `(^` n_1 `...`n_k`)`: All terms in a formula but $n_1, \ldots, n_k$.
- Location references:
  - `(!` *fnum* `l|r` i_1`...`i_n`)`: Term in formula *fnum*, `l`eft- or `r`ight-hand side, at recursive path location $i_1 \ldots i_k$.

Examples

```
{-1}  x * r + y * r + 1 >= r - 1
  |--------
{1}   r = y * 2 * x + 1

Rule? (swap-rel -1)

{-1}  r - 1 <= x * r + y * r + 1
  |--------
[1]   r = y * 2 * x + 1

Rule?  (swap! (! -1 r 1))

{-1}  r - 1 <= r * x + y * r + 1
  |--------
[1]   r = y * 2 * x + 1
```

Examples

```
{-1}  x * r + y * r + 1 >= r - 1
  |--------
{1}   r = y * 2 * x + 1

Rule? (swap-rel -1)

{-1}  r - 1 <= x * r + y * r + 1
  |--------
[1]   r = y * 2 * x + 1

Rule?  (swap! (! -1 r 1))

{-1}  r - 1 <= r * x + y * r + 1
  |--------
[1]   r = y * 2 * x + 1
```

Examples

```
{-1}  x * r + y * r + 1 >= r - 1
  |-------
{1}   r = y * 2 * x + 1

Rule? (swap-rel -1)

{-1}  r - 1 <= x * r + y * r + 1
  |-------
[1]   r = y * 2 * x + 1

Rule?  (swap! (! -1 r 1))

{-1}  r - 1 <= r * x + y * r + 1
  |-------
[1]   r = y * 2 * x + 1
```

```
{-1}  r - 1 <= r * x + y * r + 1
  |--------
[1]   r = y * 2 * x + 1

Rule? (group! (! 1 r 1) r)

[-1]  r - 1 <= r * x + y * r + 1
  |--------
{1}   r = y * (2 * x) + 1

Rule? (flip-ineq -1)

  |--------
{1}   r - 1 > r * x + y * r + 1
[2]   r = y * (2 * x) + 1
```

```
{-1}  r - 1 <= r * x + y * r + 1
  |--------
[1]   r = y * 2 * x + 1

Rule? (group! (! 1 r 1) r)

[-1]  r - 1 <= r * x + y * r + 1
  |--------
{1}   r = y * (2 * x) + 1

Rule? (flip-ineq -1)

  |--------
{1}   r - 1 > r * x + y * r + 1
[2]   r = y * (2 * x) + 1
```

```
{-1}  r - 1 <= r * x + y * r + 1
  |-------
[1]   r = y * 2 * x + 1

Rule? (group! (! 1 r 1) r)

[-1]  r - 1 <= r * x + y * r + 1
  |-------
{1}   r = y * (2 * x) + 1

Rule? (flip-ineq -1)

  |-------
{1}   r - 1 > r * x + y * r + 1
[2]   r = y * (2 * x) + 1
```

```
{-1}   r - 1 <= r * x + y * r + 1
  |-------
{1}    r = y * (2 * x) + 1

Rule? (split-ineq -1)

{-1}  r - 1 [=] r * x + y * r + 1
{-2}  r - 1 <= r * x + y * r + 1
  |-------
{1}   r = y * (2 * x) + 1

Rule? (postpone)

{-1}  r - 1 <= r * x + y * r + 1
  |-------
{1}   r - 1 [=] r * x + y * r + 1
{2}   r = y * (2 * x) + 1
```

```
{-1}   r - 1 <= r * x + y * r + 1
  |-------
{1}    r = y * (2 * x) + 1

Rule? (split-ineq -1)

{-1}  r - 1 [=] r * x + y * r + 1
{-2}  r - 1 <= r * x + y * r + 1
  |-------
{1}   r = y * (2 * x) + 1

Rule? (postpone)

{-1}  r - 1 <= r * x + y * r + 1
  |-------
{1}   r - 1 [=] r * x + y * r + 1
{2}   r = y * (2 * x) + 1
```

More Strategies

Strategy	**Description**
(`mult-by` *fnums* *term*)	Multiply formula by term
(`div-by` *fnums* *term*)	Divide formula by term
(`move-terms` *fnum* `l|r` *tnums*)	Move additive terms left and right
(`isolate` *fnum* `l|r` *tnum*)	Isolate additive terms
(`cross-mult` *fnums*)	Perform cross-multiplications
(`factor` *fnums*)	Factorize formulas
(`factor!` *exprloc*)	Factorize terms
(`mult-eq` *fnum* *fnum*)	Multiply equalities
(`mult-ineq` *fnum* *fnum*)	Multiply inequalities

More Examples

```
{-1}  (x * r + y) / pa > (r - 1) / pb
  |-------
{1}   r - y * 2 * x = 1

Rule? (cross-mult -1)

{-1}  pb * r * x + pb * y > pa * r - pa
  |-------
[1]   r - y * 2 * x = 1

Rule? (isolate 1 1 1)

[-1]  pb * r * x + pb * y > pa * r - pa
  |-------
{1}   r = 1 + y * 2 * x
```

More Examples

```
{-1}  (x * r + y) / pa > (r - 1) / pb
  |-------
{1}   r - y * 2 * x = 1

Rule? (cross-mult -1)

{-1}  pb * r * x + pb * y > pa * r - pa
  |-------
[1]   r - y * 2 * x = 1

Rule? (isolate 1 l 1)

[-1]  pb * r * x + pb * y > pa * r - pa
  |-------
{1}   r = 1 + y * 2 * x
```

More Examples

```
{-1}  (x * r + y) / pa > (r - 1) / pb
  |-------
{1}   r - y * 2 * x = 1

Rule? (cross-mult -1)

{-1}  pb * r * x + pb * y > pa * r - pa
  |-------
[1]   r - y * 2 * x = 1

Rule? (isolate 1 l 1)

[-1]  pb * r * x + pb * y > pa * r - pa
  |-------
{1}   r = 1 + y * 2 * x
```

```
{-1}  x * y - pa + na < x * na * pa
{-2}  r - y * 2 * x = 1
  |-------
{1}   2 * pa = 2 * x + 2 * y

Rule? (move-terms -1 l (2 3))

{-1}  x * y < x * na * pa + pa - na
[-2]  r - y * 2 * x = 1
  |-------
[1]   2 * pa = 2 * x + 2 * y

Rule? (factor 1)

[-1]  x * y < x * na * pa + pa - na
[-2]  r - y * 2 * x = 1
  |-------
{1}   2 * pa = 2 * (x + y)
```

```
{-1}  x * y - pa + na < x * na * pa
{-2}  r - y * 2 * x = 1
  |-------
{1}   2 * pa = 2 * x + 2 * y

Rule? (move-terms -1 l (2 3))

{-1}  x * y < x * na * pa + pa - na
[-2]  r - y * 2 * x = 1
  |-------
[1]   2 * pa = 2 * x + 2 * y

Rule? (factor 1)

[-1]  x * y < x * na * pa + pa - na
[-2]  r - y * 2 * x = 1
  |-------
{1}   2 * pa = 2 * (x + y)
```

```
{-1}  x * y - pa + na < x * na * pa
{-2}  r - y * 2 * x = 1
  |-------
{1}   2 * pa = 2 * x + 2 * y

Rule? (move-terms -1 l (2 3))

{-1}  x * y < x * na * pa + pa - na
[-2]  r - y * 2 * x = 1
  |-------
[1]   2 * pa = 2 * x + 2 * y

Rule? (factor 1)

[-1]  x * y < x * na * pa + pa - na
[-2]  r - y * 2 * x = 1
  |-------
{1}   2 * pa = 2 * (x + y)
```

```
[-1]  x * y < x * na * pa + pa - na
[-2]  r - y * 2 * x = 1
  |-------
{1}   2 * pa = 2 * (x + y)

Rule? (mult-eq -1 -2)

{-1}  (x*y)*(r-y*2*x) < (x*n*pa+pa-na)*1
[-2]  x * y < x * na * pa + pa - na
[-3]  r - y * 2 * x = 1
  |-------
[1]   2 * pa = 2 * (x + y)

Rule? (mult-ineq -1 -2 (+ +))

{-1} ((x*y)*(r-y*2*x))*(x*y)<((x*na*pa+pa-na)*1)*(x*na*pa+pa-na)
...
  |-------
[1]   2 * pa = 2 * (x + y)
```

```
[-1]  x * y < x * na * pa + pa - na
[-2]  r - y * 2 * x = 1
  |-------
{1}   2 * pa = 2 * (x + y)

Rule? (mult-eq -1 -2)

{-1}  (x*y)*(r-y*2*x) < (x*n*pa+pa-na)*1
[-2]  x * y < x * na * pa + pa - na
[-3]  r - y * 2 * x = 1
  |-------
[1]   2 * pa = 2 * (x + y)

Rule? (mult-ineq -1 -2 (+ +))

{-1} ((x*y)*(r-y*2*x))*(x*y)<((x*na*pa+pa-na)*1)*(x*na*pa+pa-na)
...
  |-------
[1]   2 * pa = 2 * (x + y)
```

```
[-1]  x * y < x * na * pa + pa - na
[-2]  r - y * 2 * x = 1
  |-------
{1}   2 * pa = 2 * (x + y)

Rule? (mult-eq -1 -2)

{-1}  (x*y)*(r-y*2*x) < (x*n*pa+pa-na)*1
[-2]  x * y < x * na * pa + pa - na
[-3]  r - y * 2 * x = 1
  |-------
[1]   2 * pa = 2 * (x + y)

Rule? (mult-ineq -1 -2 (+ +))

{-1} ((x*y)*(r-y*2*x))*(x*y)<((x*na*pa+pa-na)*1)*(x*na*pa+pa-na)
...
  |-------
[1]   2 * pa = 2 * (x + y)
```

```
...
  |-------
[1]   2 * pa = 2 * (x + y)

Rule? (div-by 1 "2")


...
  |-------
{1}   pa = (x + y)

Rule? (mult-by 1 "100")


...
  |-------
{1}   100*pa = 100*(x + y)
```

```
...
  |-------
[1]   2 * pa = 2 * (x + y)

Rule? (div-by 1 "2")

...
  |-------
{1}   pa = (x + y)

Rule? (mult-by 1 "100")

...
  |-------
{1}   100*pa = 100*(x + y)
```

```
...
  |-------
[1]   2 * pa = 2 * (x + y)

Rule? (div-by 1 "2")

...
  |-------
{1}   pa = (x + y)

Rule? (mult-by 1 "100")

...
  |-------
{1}   100*pa = 100*(x + y)
```

Field

- Field is a PVS package for simplifications in the closed field of real numbers.
- `http://shemesh.larc.nasa.gov/people/cam/Field`.
- The package consists of:
  - The strategy `field`.
  - Several *extra-tegies*.

field

```
{-1}  vox > 0
{-2}  s * s - D*D > D
{-3}  s * vix * voy - s * viy * vox /= 0
{-4}  ((s * s - D*D) * voy - D * vox * sqrt(s*s - D*D))/
       (s * (vix * voy - vox * viy)) * s * vox /= 0
{-5}   voy * sqrt(s * s - D*D) - D * vox /= 0
  |-------
{1}  (viy * sqrt(s * s - D*D) - vix * D) /
       (voy * sqrt(s * s - D*D) - vox * D) =
       (D*D - s * s) / (((s * s - D*D) * voy - D * vox *
       sqrt(s * s - D*D)) /
       (s * (vix * voy - vox * viy)) * s * vox) +
       vix / vox

Rule? (field 1)
Q.E.D.
```

field

```
{-1}  vox > 0
{-2}  s * s - D*D > D
{-3}  s * vix * voy - s * viy * vox /= 0
{-4}  ((s * s - D*D) * voy - D * vox * sqrt(s*s - D*D))/
       (s * (vix * voy - vox * viy)) * s * vox /= 0
{-5}   voy * sqrt(s * s - D*D) - D * vox /= 0
  |-------
{1}  (viy * sqrt(s * s - D*D) - vix * D) /
       (voy * sqrt(s * s - D*D) - vox * D) =
       (D*D - s * s) / (((s * s - D*D) * voy - D * vox *
       sqrt(s * s - D*D)) /
       (s * (vix * voy - vox * viy)) * s * vox) +
       vix / vox

Rule? (field 1)
Q.E.D.
```

Some Extra-tegies

Strategy	Description
(`grind-reals`)	`grind` with `real_props`
(`cancel-by` *fnum* *term*)	Cancel a common term in a formula
(`skoletin` *fnum*)	Skolemize let-in expressions
(`skeep` *fnum*)	Skolemize with same variable names
(`neg-formula` *fnum*)	Negate a formula
(`add-formula` *fnum* *fnum*)	Add formulas
(`sub-formula` *fnum* *fnum*)	Subtract formulas

grind-reals

```
  |-------
{1}   (x - 1 / 2) * (x - 1 / 2) >= 0

Rule? (grind-reals :nodistrib 1)

Q.E.D.
```

grind-reals

```
  |-------
{1}   (x - 1 / 2) * (x - 1 / 2) >= 0

Rule? (grind-reals :nodistrib 1)

Q.E.D.
```

cancel-by

```
{-1}  4 * (pa * pb) + (pa * 6) * pa = pa * ((c + 1) * 2)
  |--------
{1}   2 * pb + 3 * pa = c

Rule? (cancel-by -1 "2*pa")

{-1}  (3 * pa) + (2 * pb) = 1 + c
  |--------
{1}   2 * pa = 0
{2}   3 * pa + 2 * pb = c
```

cancel-by

```
{-1}  4 * (pa * pb) + (pa * 6) * pa = pa * ((c + 1) * 2)
  |-------
{1}   2 * pb + 3 * pa = c

Rule? (cancel-by -1 "2*pa")

{-1}  (3 * pa) + (2 * pb) = 1 + c
  |-------
{1}   2 * pa = 0
{2}   3 * pa + 2 * pb = c
```

PVS's Let-in Expressions

- Let-in expressions are used in PVS to introduce local definitions.
- They are automatically unfolded by the theorem prover.

```
  |-------
{1}   LET a = (x + 1), b = a * a, c = b * b IN c * c >= a

Rule? (assert)

  |-------
{1} 1 + x + (x*x*x*x*x*x*x*x + x*x*x*x*x*x*x)
        + (x*x*x*x*x*x*x + x*x*x*x*x*x)
        + (x*x*x*x*x*x*x + x*x*x*x*x*x)
      ...
        + (x*x + x)
        + (x*x + x)
        + (x*x + x)
        >= 1 + x
```

PVS's Let-in Expressions

- Let-in expressions are used in PVS to introduce local definitions.
- They are automatically unfolded by the theorem prover.

```
  |-------
{1}   LET a = (x + 1), b = a * a, c = b * b IN c * c >= a

Rule? (assert)

  |-------
{1} 1 + x + (x*x*x*x*x*x*x + x*x*x*x*x*x)
        + (x*x*x*x*x*x + x*x*x*x*x)
        + (x*x*x*x*x*x + x*x*x*x*x)
      ...
        + (x*x + x)
        + (x*x + x)
        + (x*x + x)
        >= 1 + x
```

skoletin

```
  |-------
{1}  LET a = (x + 1), b = a * a, c = b * b IN c * c >= a

Rule? (skoletin 1)
{-1}  a = (x + 1)
  |-------
{1}   LET b = a * a, c = b * b IN c * c >= a

Rule? (skoletin* 1)
{-1}  c = b * b
{-2}  b = a * a
[-3]  a = (x + 1)
  |-------
{1}   c * c >= a
```

skoletin

```
  |-------
{1}  LET a = (x + 1), b = a * a, c = b * b IN c * c >= a

Rule? (skoletin 1)
{-1}  a = (x + 1)
  |-------
{1}   LET b = a * a, c = b * b IN c * c >= a

Rule? (skoletin* 1)
{-1}  c = b * b
{-2}  b = a * a
[-3]  a = (x + 1)
  |-------
{1}   c * c >= a
```

skoletin

```
  |-------
{1}  LET a = (x + 1), b = a * a, c = b * b IN c * c >= a

Rule? (skoletin 1)
{-1}  a = (x + 1)
  |-------
{1}   LET b = a * a, c = b * b IN c * c >= a

Rule? (skoletin* 1)
{-1}  c = b * b
{-2}  b = a * a
[-3]  a = (x + 1)
  |-------
{1}   c * c >= a
```

More examples

```
  |-------
{1}   FORALL (nnx: nnreal, x: real):
        nnx > x - nnx*nnx AND x + 2 * nnx*nnx >= 4 * nnx
        IMPLIES nnx > 1

Rule? (skeep)
{-1}  nnx > x - nnx*nnx
{-2}  x + 2 *nnx*nnx >= 4 * nnx
  |-------
{1}   nnx > 1

Rule? (neg-formula -1)
{-1}  nnx*nnx - x > -nnx
[-2]  x + 2 * nnx*nnx >= 4 * nnx
  |-------
[1]   nnx > 1
```

More examples

```
  |-------
{1}   FORALL (nnx: nnreal, x: real):
        nnx > x - nnx*nnx AND x + 2 * nnx*nnx >= 4 * nnx
        IMPLIES nnx > 1

Rule? (skeep)
{-1}  nnx > x - nnx*nnx
{-2}  x + 2 *nnx*nnx >= 4 * nnx
  |-------
{1}   nnx > 1

Rule? (neg-formula -1)
{-1}  nnx*nnx - x > -nnx
[-2]  x + 2 * nnx*nnx >= 4 * nnx
  |-------
[1]   nnx > 1
```

More examples

```
  |-------
{1}   FORALL (nnx: nnreal, x: real):
        nnx > x - nnx*nnx AND x + 2 * nnx*nnx >= 4 * nnx
        IMPLIES nnx > 1

Rule? (skeep)
{-1}  nnx > x - nnx*nnx
{-2}  x + 2 * nnx*nnx >= 4 * nnx
  |-------
{1}   nnx > 1

Rule? (neg-formula -1)
{-1}  nnx*nnx - x > -nnx
[-2]  x + 2 * nnx*nnx >= 4 * nnx
  |-------
[1]   nnx > 1
```

```
{-1}  nnx*nnx - x > -nnx
[-2]  x + 2 * nnx*nnx >= 4 * nnx
  |-------
[1]   nnx > 1

Rule? (add-formulas -1 -2)
{-1}  3 * (nnx*nnx) > -nnx + 4 * nnx
  |-------
[1]   nnx > 1

Rule? (cancel-by -1 "nnx")

Q.E.D.
```

```
{-1}  nnx*nnx - x > -nnx
[-2]  x + 2 * nnx*nnx >= 4 * nnx
  |-------
[1]   nnx > 1

Rule? (add-formulas -1 -2)
{-1}  3 * (nnx*nnx) > -nnx + 4 * nnx
  |-------
[1]   nnx > 1

Rule? (cancel-by -1 "nnx")

Q.E.D.
```

```
{-1}  nnx*nnx - x > -nnx
[-2]  x + 2 * nnx*nnx >= 4 * nnx
  |-------
[1]   nnx > 1

Rule? (add-formulas -1 -2)
{-1}  3 * (nnx*nnx) > -nnx + 4 * nnx
  |-------
[1]   nnx > 1

Rule? (cancel-by -1 "nnx")

Q.E.D.
```

Advanced Strategies

Importing	**Scope**
`Sturm@strategies`	Single-varible polynomial relations
`Tarski@strategies`	Boolean expressions of polynomial relations
`Bernstein@strategies`	Multi-variable polynomial relations
`affine_arith@strategies`	Multi-variable polynomial relations (rigorous approximations)
`interval_arith@strategies`	Real-valued functions (rigorous approximations)
`exact_real_arith@strategies`	Real-valued functions (arbitrary precision)
MetiTarski	Real-valued functions (external oracle)

sturm

Decision procedure based on Sturm's theorem

```
IMPORTING Sturm@strategies

sturm_fa :
  |-------
{1}   FORALL (x: real): x - x * x <= 1 / 4
Rule? (sturm)
Q.E.D.

sturm_ex :
  |-------
{1}   EXISTS (x: real): x >= 0 AND x ^ 2 - x < 0
Rule? (sturm)
Q.E.D.
```

sturm

Decision procedure based on Sturm's theorem

```
IMPORTING Sturm@strategies

sturm_fa :
  |-------
{1}   FORALL (x: real): x - x * x <= 1 / 4
Rule? (sturm)
Q.E.D.

sturm_ex :
  |-------
{1}   EXISTS (x: real): x >= 0 AND x ^ 2 - x < 0
Rule? (sturm)
Q.E.D.
```

sturm

Decision procedure based on Sturm's theorem

```
IMPORTING Sturm@strategies

sturm_fa :
  |-------
{1}   FORALL (x: real): x - x * x <= 1 / 4
Rule? (sturm)
Q.E.D.

sturm_ex :
  |-------
{1}   EXISTS (x: real): x >= 0 AND x ^ 2 - x < 0
Rule? (sturm)
Q.E.D.
```

sturm

Decision procedure based on Sturm's theorem

```
IMPORTING Sturm@strategies

sturm_fa :
  |-------
{1}   FORALL (x: real): x - x * x <= 1 / 4
Rule? (sturm)
Q.E.D.

sturm_ex :
  |-------
{1}   EXISTS (x: real): x >= 0 AND x ^ 2 - x < 0
Rule? (sturm)
Q.E.D.
```

mono-poly

Dicharges monocity properties of polynomials

```
mono_fa :
  |-------
{1}   FORALL (x,y: real):
        x >= 1 AND x < y IMPLIES
         (x - 1/4) ^ 2 <= y*y - (1/2)*y + (1/16)

Rule? (mono-poly)
Q.E.D.


mono_ex :
  |-------
{1}   EXISTS (x,y: real): x < y AND x^2 >= sq(y)

Rule? (mono-poly)
Q.E.D.
```

mono-poly

Dicharges monocity properties of polynomials

```
mono_fa :
  |-------
{1}   FORALL (x,y: real):
        x >= 1 AND x < y IMPLIES
         (x - 1/4) ^ 2 <= y*y - (1/2)*y + (1/16)

Rule? (mono-poly)
Q.E.D.
```

```
mono_ex :
  |-------
{1}   EXISTS (x,y: real): x < y AND x^2 >= sq(y)

Rule? (mono-poly)
Q.E.D.
```

mono-poly

Dicharges monocity properties of polynomials

```
mono_fa :
  |-------
{1}   FORALL (x,y: real):
        x >= 1 AND x < y IMPLIES
         (x - 1/4) ^ 2 <= y*y - (1/2)*y + (1/16)

Rule? (mono-poly)
Q.E.D.


mono_ex :
  |-------
{1}   EXISTS (x,y: real): x < y AND x^2 >= sq(y)

Rule? (mono-poly)
Q.E.D.
```

mono-poly

Dicharges monocity properties of polynomials

```
mono_fa :
  |-------
{1}   FORALL (x,y: real):
        x >= 1 AND x < y IMPLIES
         (x - 1/4) ^ 2 <= y*y - (1/2)*y + (1/16)

Rule? (mono-poly)
Q.E.D.

mono_ex :
  |-------
{1}   EXISTS (x,y: real): x < y AND x^2 >= sq(y)

Rule? (mono-poly)
Q.E.D.
```

tarski

Decision procedure based on Tarski's theorem

```
IMPORTING Tarski@strategies

tarski_fa :

  |-------
{1} FORALL (x:real): (x-2)^2*(-x+4) > 0 AND
      x^2*(x-3)^2 >= 0 AND x-1 >= 0 AND -(x-3)^2+1 > 0
      IMPLIES -(x-11/12)^3*(x-41/10)^3 >= 0

Rule? (tarski)
Q.E.D.

tarski_ex :

  |-------
{1} EXISTS (x:real): (x-2)^2*(-x+4) > 0 AND x^2*(x-3)^2 >= 0
    AND x-1 >= 0 AND -(x-3)^2+1 > 0
    AND -(x-11/12)^3*(x-41/10)^3 < 1/10

Rule? (tarski)
Q.E.D.
```

tarski

Decision procedure based on Tarski's theorem

```
IMPORTING Tarski@strategies

tarski_fa :

  |-------
{1} FORALL (x:real): (x-2)^2*(-x+4) > 0 AND
      x^2*(x-3)^2 >= 0 AND x-1 >= 0 AND -(x-3)^2+1 > 0
      IMPLIES -(x-11/12)^3*(x-41/10)^3 >= 0

Rule? (tarski)
Q.E.D.

tarski_ex :

  |-------
{1} EXISTS (x:real): (x-2)^2*(-x+4) > 0 AND x^2*(x-3)^2 >= 0
    AND x-1 >= 0 AND -(x-3)^2+1 > 0
    AND -(x-11/12)^3*(x-41/10)^3 < 1/10

Rule? (tarski)
Q.E.D.
```

tarski

Decision procedure based on Tarski's theorem

```
IMPORTING Tarski@strategies

tarski_fa :

  |-------
{1} FORALL (x:real): (x-2)^2*(-x+4) > 0 AND
      x^2*(x-3)^2 >= 0 AND x-1 >= 0 AND -(x-3)^2+1 > 0
      IMPLIES -(x-11/12)^3*(x-41/10)^3 >= 0

Rule? (tarski)
Q.E.D.

tarski_ex :

  |-------
{1} EXISTS (x:real): (x-2)^2*(-x+4) > 0 AND x^2*(x-3)^2 >= 0
    AND x-1 >= 0 AND -(x-3)^2+1 > 0
    AND -(x-11/12)^3*(x-41/10)^3 < 1/10

Rule? (tarski)
Q.E.D.
```

tarski

Decision procedure based on Tarski's theorem

```
IMPORTING Tarski@strategies

tarski_fa :

  |-------
{1} FORALL (x:real): (x-2)^2*(-x+4) > 0 AND
      x^2*(x-3)^2 >= 0 AND x-1 >= 0 AND -(x-3)^2+1 > 0
      IMPLIES -(x-11/12)^3*(x-41/10)^3 >= 0

Rule? (tarski)
Q.E.D.

tarski_ex :

  |-------
{1} EXISTS (x:real): (x-2)^2*(-x+4) > 0 AND x^2*(x-3)^2 >= 0
    AND x-1 >= 0 AND -(x-3)^2+1 > 0
    AND -(x-11/12)^3*(x-41/10)^3 < 1/10

Rule? (tarski)
Q.E.D.
```

bernstein

Rigorous approximations using Bernstein polynomial basis

```
IMPORTING Bernstein@strategies

bernstein_fa :

  |-------
{1}  FORALL (x,y:real):
      -0.5 <= x AND x <= 1 AND -2 <= y AND y <= 1 IMPLIES
      4*x^2 - (21/10)*x^4 + (1/3)*x^6 + (x-3)*y - 4*y^2 + 4*y^4 > -3.4

Rule? (bernstein)
Q.E.D.

bernstein_ex :

  |-------
{1} EXISTS (x,y:real):
      -0.5 <= x AND x <= 1 AND -2 <= y AND y <= 1 AND
      4*x^2 - (21/10)*x^4 + (1/3)*x^6 + (x-3)*y - 4*y^2 + 4*y^4 < -3.39

Rule? (bernstein)
Q.E.D.
```

bernstein

Rigorous approximations using Bernstein polynomial basis

```
IMPORTING Bernstein@strategies

bernstein_fa :

  |-------
{1}  FORALL (x,y:real):
       -0.5 <= x AND x <= 1 AND -2 <= y AND y <= 1 IMPLIES
       4*x^2 - (21/10)*x^4 + (1/3)*x^6 + (x-3)*y - 4*y^2 + 4*y^4 > -3.4

Rule? (bernstein)
Q.E.D.

bernstein_ex :

  |-------
{1} EXISTS (x,y:real):
      -0.5 <= x AND x <= 1 AND -2 <= y AND y <= 1 AND
      4*x^2 - (21/10)*x^4 + (1/3)*x^6 + (x-3)*y - 4*y^2 + 4*y^4 < -3.39

Rule? (bernstein)
Q.E.D.
```

bernstein

Rigorous approximations using Bernstein polynomial basis

```
IMPORTING Bernstein@strategies

bernstein_fa :

  |-------
{1}  FORALL (x,y:real):
       -0.5 <= x AND x <= 1 AND -2 <= y AND y <= 1 IMPLIES
       4*x^2 - (21/10)*x^4 + (1/3)*x^6 + (x-3)*y - 4*y^2 + 4*y^4 > -3.4

Rule? (bernstein)
Q.E.D.

bernstein_ex :

  |-------
{1} EXISTS (x,y:real):
       -0.5 <= x AND x <= 1 AND -2 <= y AND y <= 1 AND
       4*x^2 - (21/10)*x^4 + (1/3)*x^6 + (x-3)*y - 4*y^2 + 4*y^4 < -3.39

Rule? (bernstein)
Q.E.D.
```

bernstein

Rigorous approximations using Bernstein polynomial basis

```
IMPORTING Bernstein@strategies

bernstein_fa :

  |-------
{1}  FORALL (x,y:real):
       -0.5 <= x AND x <= 1 AND -2 <= y AND y <= 1 IMPLIES
       4*x^2 - (21/10)*x^4 + (1/3)*x^6 + (x-3)*y - 4*y^2 + 4*y^4 > -3.4

Rule? (bernstein)
Q.E.D.

bernstein_ex :

  |-------
{1} EXISTS (x,y:real):
       -0.5 <= x AND x <= 1 AND -2 <= y AND y <= 1 AND
       4*x^2 - (21/10)*x^4 + (1/3)*x^6 + (x-3)*y - 4*y^2 + 4*y^4 < -3.39

Rule? (bernstein)
Q.E.D.
```

affine

Rigorous approximations using affine arithmetic

```
IMPORTING affine_arith@strategies

affine_fa :

  |-------
{1}  FORALL (x,y:real):
      -0.5 <= x AND x <= 1 AND -2 <= y AND y <= 1 IMPLIES
      4*x^2 - (21/10)*x^4 + (1/3)*x^6 + (x-3)*y - 4*y^2 + 4*y^4 > -3.4

Rule? (affine)
Q.E.D.

affine_ex :

  |-------
{1} EXISTS (x,y:real):
      -0.5 <= x AND x <= 1 AND -2 <= y AND y <= 1 AND
      4*x^2 - (21/10)*x^4 + (1/3)*x^6 + (x-3)*y - 4*y^2 + 4*y^4 < -3.39

Rule? (affine)
Q.E.D.
```

affine

Rigorous approximations using affine arithmetic

```
IMPORTING affine_arith@strategies

affine_fa :

  |-------
{1}  FORALL (x,y:real):
       -0.5 <= x AND x <= 1 AND -2 <= y AND y <= 1 IMPLIES
       4*x^2 - (21/10)*x^4 + (1/3)*x^6 + (x-3)*y - 4*y^2 + 4*y^4 > -3.4

Rule? (affine)
Q.E.D.

affine_ex :

  |-------
{1} EXISTS (x,y:real):
       -0.5 <= x AND x <= 1 AND -2 <= y AND y <= 1 AND
       4*x^2 - (21/10)*x^4 + (1/3)*x^6 + (x-3)*y - 4*y^2 + 4*y^4 < -3.39

Rule? (affine)
Q.E.D.
```

affine

Rigorous approximations using affine arithmetic

```
IMPORTING affine_arith@strategies

affine_fa :

  |-------
{1}  FORALL (x,y:real):
      -0.5 <= x AND x <= 1 AND -2 <= y AND y <= 1 IMPLIES
      4*x^2 - (21/10)*x^4 + (1/3)*x^6 + (x-3)*y - 4*y^2 + 4*y^4 > -3.4

Rule? (affine)
Q.E.D.

affine_ex :

  |-------
{1} EXISTS (x,y:real):
      -0.5 <= x AND x <= 1 AND -2 <= y AND y <= 1 AND
      4*x^2 - (21/10)*x^4 + (1/3)*x^6 + (x-3)*y - 4*y^2 + 4*y^4 < -3.39

Rule? (affine)
Q.E.D.
```

affine

Rigorous approximations using affine arithmetic

```
IMPORTING affine_arith@strategies

affine_fa :

  |-------
{1}  FORALL (x,y:real):
      -0.5 <= x AND x <= 1 AND -2 <= y AND y <= 1 IMPLIES
      4*x^2 - (21/10)*x^4 + (1/3)*x^6 + (x-3)*y - 4*y^2 + 4*y^4 > -3.4

Rule? (affine)
Q.E.D.

affine_ex :

  |-------
{1} EXISTS (x,y:real):
      -0.5 <= x AND x <= 1 AND -2 <= y AND y <= 1 AND
      4*x^2 - (21/10)*x^4 + (1/3)*x^6 + (x-3)*y - 4*y^2 + 4*y^4 < -3.39

Rule? (affine)
Q.E.D.
```

aa-numerical

Numerical approximations using affine arithmetic

```
{-1}  -0.5 <= x
{-2}  x <= 1
{-3}  -2 <= y
{-4}  y <= 1
  |-------
{1} 4*x^2-(21/10)*x^4+(1/3)*x^6+(x-3)*y-4*y^2+4*y^4 > -3.4

Rule? (aa-numerical (! 1 1) :precision 5)

{-1} 4*x^2-(21/10)*x^4+(1/3)*x^6+(x-3)*y-4*y^2 + 4*y^4
       ## [|-3.43158, 55.90987|]
...
  |-------
...
```

aa-numerical

Numerical approximations using affine arithmetic

```
{-1}  -0.5 <= x
{-2}  x <= 1
{-3}  -2 <= y
{-4}  y <= 1
  |-------
{1} 4*x^2-(21/10)*x^4+(1/3)*x^6+(x-3)*y-4*y^2+4*y^4 > -3.4

Rule? (aa-numerical (! 1 l) :precision 5)

{-1} 4*x^2-(21/10)*x^4+(1/3)*x^6+(x-3)*y-4*y^2 + 4*y^4
       ## [|-3.43158, 55.90987|]
...
  |-------
...
```

interval

Rigorous approximations using interval arithmetic

```
IMPORTING interval_arith@strategies

sin_x_cos :

  |-------
{1}   EXISTS (d: real):
        d ## [|0, 90|] AND sin(d*pi/180)*cos(d*pi/180) <= 1/2

Rule? (interval)
Q.E.D.

tr_200_250_abs_35 :

{-1}  abs(phi) <= 35
{-2}  v ## [|200, 250|]
  |-------
{1}   abs(((180*g)/(pi*v*0.514))*tan((pi*phi)/180)) <= 3.825

Rule? (interval)
Q.E.D.
```

interval

Rigorous approximations using interval arithmetic

```
IMPORTING interval_arith@strategies

sin_x_cos :

  |-------
{1}   EXISTS (d: real):
        d ## [|0, 90|] AND sin(d*pi/180)*cos(d*pi/180) <= 1/2

Rule? (interval)
Q.E.D.

tr_200_250_abs_35 :

{-1}  abs(phi) <= 35
{-2}  v ## [|200, 250|]
  |-------
{1}   abs(((180*g)/(pi*v*0.514))*tan((pi*phi)/180)) <= 3.825

Rule? (interval)
Q.E.D.
```

interval

Rigorous approximations using interval arithmetic

```
IMPORTING interval_arith@strategies

sin_x_cos :

  |-------
{1}   EXISTS (d: real):
        d ## [|0, 90|] AND sin(d*pi/180)*cos(d*pi/180) <= 1/2

Rule? (interval)
Q.E.D.

tr_200_250_abs_35 :

{-1}  abs(phi) <= 35
{-2}  v ## [|200, 250|]
  |-------
{1}   abs(((180*g)/(pi*v*0.514))*tan((pi*phi)/180)) <= 3.825

Rule? (interval)
Q.E.D.
```

interval

Rigorous approximations using interval arithmetic

```
IMPORTING interval_arith@strategies

sin_x_cos :

  |-------
{1}   EXISTS (d: real):
        d ## [|0, 90|] AND sin(d*pi/180)*cos(d*pi/180) <= 1/2

Rule? (interval)
Q.E.D.

tr_200_250_abs_35 :

{-1}  abs(phi) <= 35
{-2}  v ## [|200, 250|]
  |-------
{1}   abs(((180*g)/(pi*v*0.514))*tan((pi*phi)/180)) <= 3.825

Rule? (interval)
Q.E.D.
```

numerical

Numerical approximations using interval arithmetic

```
{-1} abs(phi) <= 35
{-2} v ## [|200, 250|]
  |-------
{1} abs(((180*g)/(pi*v*0.514))*tan((pi*phi)/180)) <= 3.825

Rule? (numerical (! 1 l) :precision 5)

{-1} abs(((180*g)/(pi*v*0.514))*tan((pi*phi)/180)) ##
       [|0, 3.82457|]
...
  |-------
...
```

numerical

Numerical approximations using interval arithmetic

```
{-1} abs(phi) <= 35
{-2} v ## [|200, 250|]
  |-------
{1} abs(((180*g)/(pi*v*0.514))*tan((pi*phi)/180)) <= 3.825

Rule? (numerical (! 1 l) :precision 5)

{-1} abs(((180*g)/(pi*v*0.514))*tan((pi*phi)/180)) ##
       [|0, 3.82457|]
...
  |-------
...
```

era-numerical

Exact real arithmetic

```
IMPORTING exact_real_arith@strategies

sqrt_pi :

  |-------
{1}   sqrt(pi) < 2

Rule? (era-numerical (! 1 l) :precision 20)

{-1}  sqrt(pi) < 1.7724538509055160273
{-2}  1.7724538509055160272 < sqrt(pi)
  |-------
{1}   sqrt(pi) < 2
```

era-numerical

Exact real arithmetic

```
IMPORTING exact_real_arith@strategies

sqrt_pi :

  |-------
{1}   sqrt(pi) < 2

Rule? (era-numerical (! 1 l) :precision 20)

{-1}  sqrt(pi) < 1.77245385090551602731
{-2}  1.77245385090551602729 < sqrt(pi)
  |-------
{1}   sqrt(pi) < 2
```

metit

Using MetiTarski as an external oracle

```
Ayad_Marche :

  |-------
{1}   FORALL (r: real): abs(r) <= 1 IMPLIES
         abs(0.9890365552+1.130258690*r+0.5540440796*r*r-exp(r))
          <= (1-2^-16)*2^-4

Rule? (metit)
MetiTarski Input = fof(pvs2metit,conjecture, (![R1]: ((abs(R1) <= 1)
=> (abs(((((9890365552 / 10000000000) + ((1130258690 / 1000000000) *
R1)) + (((5540440796 / 10000000000) * R1) * R1)) - exp(R1))) <=
((1 - 2^-16) * 2^-4))))).
SZS status Theorem for Ayad_Marche.tptp
Processor time: 0.081 = 0.048 (Metis) + 0.033 (RCF)
Maximum weight in proof search: 424
MetiTarski succesfully proved.
Trusted oracle: MetiTarski.
Q.E.D.
```

metit

Using MetiTarski as an external oracle

```
Ayad_Marche :

  |-------
{1}   FORALL (r: real): abs(r) <= 1 IMPLIES
        abs(0.9890365552+1.130258690*r+0.5540440796*r*r-exp(r))
         <= (1-2^-16)*2^-4

Rule? (metit)
MetiTarski Input = fof(pvs2metit,conjecture, (![R1]: ((abs(R1) <= 1)
=> (abs(((((9890365552 / 10000000000) + ((1130258690 / 1000000000) *
R1)) + (((5540440796 / 10000000000) * R1) * R1)) - exp(R1))) <=
((1 - 2^-16) * 2^-4))))).
SZS status Theorem for Ayad_Marche.tptp
Processor time: 0.081 = 0.048 (Metis) + 0.033 (RCF)
Maximum weight in proof search: 424
MetiTarski succesfully proved.
Trusted oracle: MetiTarski.
Q.E.D.
```