REAL AUTOMATION IN THE FIELD[*]

CÉSAR MUÑOZ[†] AND MICAELA MAYERO[‡]

Abstract. We provide a package of strategies for automation of non-linear arithmetic in PVS. In particular, we describe a simplification procedure for the field of real numbers and a strategy for cancellation of common terms.

Key words. Non-linear arithmetic, PVS strategies, real number automation

Subject classification. Computer Science

1. Introduction. While conducting research on the formal safety analysis of Air Traffic Management (ATM) systems at the NASA Langley Research Center [2, 3, 6], we have found the verification system PVS [7] very well suited for specifying *avionics systems*. Avionics systems are *hybrid systems*, that is, they are digital systems that interact with the continuous physical environment where they are deployed. These two levels of dynamics, discrete and continuous, can be easily modeled using PVS higher-order logic. Despite that, deductive methods, such as theorem proving, are not usually the first approach for modeling and verification of hybrid systems. Most of the literature on verification of hybrid systems concerns variations of algorithmic methods, such as model checking, on restricted sub-classes of the general problem. This can be explained by the fact that although theorem provers based on higher-order logic, e.g., PVS [7], Coq [1], Isabelle-HOL [8], integrate expressive specification languages with powerful theorem provers, they lack of adequate support for continuous mathematics.

The PVS system, for instance, incorporates a rich type system supporting predicate sub-typing and dependent types. It also comes with a set of decision procedures that solve problems in a broad range of decidable fields. Moreover, it includes in the prelude library a large collection of properties for real numbers (for example, `real_props` and `real_axioms`) that are applied automatically by the decision procedures. This is true except for non-linear ones, i.e., those involving multiplication or division of non-numeric terms. Indeed, the PVS command `(GRIND :theories "real_props")`, although useful in several cases, fails to discharge a simple non-linear formula such as

$$\frac{a+1}{\frac{a+1}{b+1}} - b = 1 \tag{1.1}$$

where a, b are positive real numbers. In [9], Rushby and Shankar show how a small augmentation of the PVS prelude library and a deep knowledge of the PVS rewriting mechanism can provide effective automatic arithmetic simplification. However, a lack of good documentation of PVS rewriting features, forbid the access of all these capabilities to a normal user.

Manual proofs of formulas involving non-linear arithmetic can be tedious. It usually requires the ability to recognize by heart the names of the lemmas in the prelude library. It may help that some names in this

[*]This work was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-97046 while the authors were in residence at ICASE, NASA Langley Research Center, Hampton, VA 23681-2199, USA.

[†]Main and contact Author: ICASE, Mail Stop 132C, NASA Langley Research Center, Hampton, VA 23681-2199, USA, e-mail: `munoz@icase.edu`.

[‡]INRIA, Domaine de Voluceau - Rocquencourt - B.P. 105, 78153 Le Chesnay Cedex, France, e-mail: `Micaela.Mayero@inria.fr`.

library are mnemonics. For example, to check Formula 1.1 we could use Lemma `div_div1`, which states

$$\frac{x}{\frac{n0y}{n0z}} = \frac{x \times n0z}{n0y}$$

where $n0y$ and $n0z$ denote real numbers different from zero. After applying this lemma on Formula 1.1, via `(REWRITE "div_div1")`, we get the formula

$$\frac{1 + b + (a \times b + a)}{1 + a} - b = 1 \tag{1.2}$$

At this point, we may want to rearrange Formula 1.2. In general, this is done with the proof command `CASE`. However, in this simple example, we add b to both sides of the equality using the command `BOTH-SIDES`, and finish the proof with `(ASSERT)`.

The complete proof of Formula 1.1 is a couple of lines long and none of the steps is really difficult. However, the steps become the more a more involved when checking large formulas. Consider for example Formula 1.3, taken from the verification of a conflict detection and resolution algorithm developed at ICASE - NASA Langley [6]. The first naive attempt to verify this formula, under the assumption that it is well-defined, was over a hundred of lines.

$$\frac{v_{iy} \times \sqrt{s^2 - D^2} - v_{ix} \times D}{v_{oy} \times \sqrt{s^2 - D^2} - v_{ox} \times D} = \frac{D^2 - s^2}{\frac{(s^2-D^2) \times v_{oy} - D \times v_{ox} \times \sqrt{s^2-D^2}}{s \times (v_{ix} \times v_{oy} - v_{ox} \times v_{iy})} \times s \times v_{ox}} + v_{ix}/v_{ox} \tag{1.3}$$

In [11], Di Vito presents a package of strategies and functions for manipulating arithmetic expressions in PVS. Using this package, we have developed a simplification procedure for the field of real numbers. The strategy, called `FIELD`, helped us to reduce the proofs of formulas 1.1, 1.2, and 1.3 to just one line. In Section 2, we describe the strategy and give some examples of usage. In addition to `FIELD`, we have developed strategies for cancellation of common terms, and some support strategies for arithmetic manipulation. These strategies are described in Section 3 and Section 4. Finally, in Section 5, we discuss some issues related to real automation in PVS and we summarize our work. The package is available at:
`http://www.icase.edu./~munoz/Field/field.html`.

Most of the strategies developed in our package act on *relational formulas*, that is, arithmetic formulas having the form $e \lesseqqgtr f$, where $\lesseqqgtr \in \{=, <, \leq, >, \geq\}$. Through this paper, for the sake of readability, we have used a LaTeX-style for writing PVS code.

2. Reals in the Field. `FIELD` is a simplification procedure for relational formulas on the field of real numbers. It was originally based on the tactic `Field` of Coq V7 [5], but it has been extensively adapted to cope with PVS idiosyncrasies. The tactic `Field` of Coq is *reflexive*, i.e., it uses Coq's own reduction mechanism to avoid rewriting steps, which yield large proof terms in the system. The tactic `Field` first translates the formula to an intermediate representation (the reflexive step), then removes all the inverses appearing in the original formula, and finally applies `Ring`, another reflexive tactic that implements a decision procedure for the ring of real numbers. In contrast, the strategy `FIELD` of PVS largely depends on decision procedures, which implement `Ring` capabilities but are not reflexive. It also has been extended to handle inequalities.

2.1. Algorithm. `FIELD`, applied to the relational formula $e \lesseqqgtr f$, is described by the algorithm:

1. Collect all the inverses $x_1, \ldots, x_n$ appearing as denominators of sub-terms of e, f. If $n = 0$, apply `ASSERT`.
2. Assume $p \neq 0$, where $p = x_1 \times \ldots \times x_n$.

3. Let $e' = p \times e$ and $f' = p \times f$.
4. If $\lesseqqgtr$ is the equality operator, replace the equality $e = f$ with the equality $e' = f'$.
5. If $\lesseqqgtr$ is in $\{<, \leq, >, \geq\}$, replace the inequality $e \lesseqqgtr f$ with the two inequalities (1) $e' \lesseqqgtr f'$, assuming $p > 0$, and (2) $f' \lesseqqgtr e'$, assuming $p < 0$.
6. For each one of the new formulas:
   (a) Cancel inverses in e', f', and simplify with decision procedures, i.e., `ASSERT` and `GRIND`.
   (b) Call `FIELD` with the resulting formula.

As explained in [5], the recursive call is needed to handle the case of nested divisions. In order to remove inverses, an explicit control on the application of distributive laws is required. Unfortunately, PVS applies simplification rules, including distributive laws, after performing any rewriting step. Note, for example, that the rewrite command on Formula 1.1 yields a rearranged Formula 1.2 where multiplications have been completely distributed. In particular, the common term $(a+1)$ is no longer available for cancellation. Below, we describe a strategy that blocks the automatic application of distribute laws by introducing new variable names.

2.2. Use and Examples. Following is the description of the strategy and some examples of usage. The examples refer to the PVS code in Figure 2.1.

syntax: `(FIELD &OPTIONAL fnum[1] grind?[T])`

effect: Applies a simplification procedure for the field of real numbers on the formula `fnum`.

requirements: If `fnum` is a formula in the consequent, it should have the form $\forall(x_1, \ldots, x_n) : e \lesseqqgtr f$. Otherwise, it should have the form $\exists(x_1, \ldots, x_n) : e \lesseqqgtr f$. In both cases $n \geq 0$. `FIELD` uses `GRIND` to prove that denominators are different from zero. That strategy sometimes falls in an infinite loop. Using the key `grind?` $=$ `NIL` is safer, but less effective.

The command `(FIELD)` takes care of lemmas `ex1` and `ex2`, which correspond to formulas 1.1 and 1.2, respectively. `FIELD` also works on inequalities, e.g., the sequence of proof commands `(SKOSIMP)(FIELD)` discharges Lemma `ex3`.

Validation of Formula 1.3 requires a little bit more work. First, we import the theory `reals@sqrt`, which makes part of the NASA Langley PVS library `reals`. This library is available at
`http://shemesh.larc.nasa.gov/ftp/larc/PVS-library/pvslib.html`.
Finally, we write Formula 1.3 as Lemma `kb3d` in PVS and prove it with the sequence of commands `(SKOSIMP)(FIELD 4)`. Only one TCC of `kb3d` has to be proved by hand; it can be easily discharged using the properties in the theory `sqrt`.

Finally, `FIELD` can also be used to remove inverses. For example, the sequence of proof commands `(SKOLEM 1 ("nx" "ny" "x" "y"))(FLATTEN)(FIELD -1)` on Lemma `ex4` yields

```
{-1}  x × nx - y × nx = 0
  |-------
[1]   x = y
```

To finish this proof, we multiply formula 1 with `nx`, for example using Di Vito's strategy [11] `MULT-BY`, and then apply the proof command `(ASSERT)`. We may also recognize that the term `nx` in formula `{-1}` can be cancelled. We show a strategy for cancellation of common terms in the next section.

3. Cancellation of Common Terms. `FIELD` is very effective removing inverses, however, by doing that, it tends to introduce common terms as in the case of Lemma `ex4` above. `CANCEL-BY` is a simplification strategy for cancellation of a common term appearing in a relational formula. First, it divides both sides by the common term, then it distributes, and finally, it simplifies the expressions using PVS decision procedures.

```
a,b   : VAR posreal
x,y   : VAR real
nx,ny : VAR nzreal

ex1 : LEMMA (a+1)/((a+1)/(b+1)) - b = 1

ex2 : LEMMA (1+b+(a×b+a))/(1+a) - b = 1

ex3 : LEMMA x > 1 ⇒
  x/((x-1)×(x+1)) - 1 /((x-1)×(x+1)) ≥ 1 /(x+1) - 1

ex4 : LEMMA x × nx/ny - y × nx/ny = 0 ⇒
  x = y

ex5 : LEMMA 4×((a+1)×b) + ((a+1)×6)×(a+1) = -(a+1)×((x+1)×2) ⇒
  2×b + 3×(a+1) + x + 1  = 0

IMPORTING reals@sqrt
t,vix,viy,vox,voy,s : VAR real
D                   : VAR posreal
kb3d : LEMMA
  vox > 0 ∧ s > D ∧ s × vix × voy - s × viy × vox ≠ 0 ∧
  ((s × s - D × D) × voy - D × vox × sqrt(s × s - D × D)) /
   (s × (vix × voy - vox × viy)) × s × vox ≠ 0 ∧
  voy × sqrt(s × s - D × D) - D × vox ≠ 0
  ⇒
    (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
```

Fig. 2.1. *PVS code for examples of* `FIELD`

3.1. Algorithm. `CANCEL-BY`, applied to a term t in a relational formula $e \lesseqgtr f$, is described by the algorithm:

1. Assume $t \neq 0$.
2. If $\lesseqgtr$ is the equality operator, replace the equality $e = f$ with the equality $e/t = f/t$.
3. If $\lesseqgtr$ is in $\{<, \leq, >, \geq\}$, replace the inequality $e \lesseqgtr f$ with the two inequalities (1) $e/t \lesseqgtr f/t$, assuming $t > 0$, and (2) $f/t \lesseqgtr e/t$, assuming $t < 0$.
4. For each one of the new formulas:
    (a) Distribute t on expressions e and f. Let e' and f' be the new formulas.

(b) Remove inverses in e', f', and simplify with decision procedures.

3.2. Use and Examples. Following is the description of the strategy and some examples of usage.

syntax: `(CANCEL-BY fnum expr)`

effect: Cancels the expression `expr` in the formula `fnum`.

The sequent

```
{-1}  x × nx - y × nx = 0
  |-------
[1]   x = y
```

coming from Lemma `ex4`, Figure 2.1, can be discharged with the command `(CANCEL-BY -1 "nx")`.

The commands `(SKOLEM 1 ("a" "b" "x")) (FLATTEN)` on Lemma `ex5`, Figure 2.1, yield the sequent

```
{-1}  4 × ((a + 1) × b) + ((a + 1) × 6) × (a + 1) = -(a + 1) × ((x + 1) × 2)
  |-------
{1}   2 × b + 3 × (a + 1) + x + 1 = 0
```

It can be discharge with the command `(CANCEL-BY -1 "(a+1)×2")`.

4. "Extra-tegies". In addition to `FIELD` and `CANCEL-BY`, we have implemented some general strategies, in a package called `extra-tegies`, to support automation on real numbers and arithmetic manipulations. This package is complementary to the one developed by Di Vito [11].

4.1. Automation support.

syntax: `(GRIND-REALS)`

effect: Applies PVS command `GRIND` with the theories `real_props` and `extra_real_props`. The latter one makes part of the package in [11]. `GRIND-REALS` does no expand definitions.

caveat: Just as `GRIND`, `GRIND-REALS` may not terminate.

syntax: `(REALS-PROPS &OPTIONAL fnums[*])`

effect: Applies PVS command `AUTO-REWRITE`, with the theories `real_props` and `extra_real_props`, on the formulas `fnums`.

tip: If `GRIND-REALS` does too much, try `REAL-PROPS`.

syntax: `(NAME-DISTRIBS &OPTIONAL name fnums[*])`

effect: Introduces new names, based on `name`, to block the automatic application of distribute laws in formulas `fnums`. If no name is given, new names are automatically generated.

usage: For illustration purposes, consider the sequent

```
{-1}  x×(x + 1) = y
{-2}  y×(y + 1) = x
  |-------
{1}   x×(x + 2) = y×(y + 2)
```

the command `(NAME-DISTRIBS "A" 1)` uses "A" to create new names:

```
{-1,(A1:)}
      (y + 2) = A12__
{-2,(A1:)}
      (x + 2) = A11__
[-3]  x × (x + 1) = y
[-4]  y × (y + 1) = x
```

```
  |-------
{1}   x × A11__ = y × A12__
```

In contrast, `(NAME-DISTRIBS :name NIL :fnums 1)` generates new names each time the command is invoked.

caveat: If `name` is null, there is no guarantee that the same names are used when re-running the proof. In that case, explicit reference to expressions containing new names should be avoided. If `name` is supplied, the new names are guaranteed to be the same each time the proof is executed.

syntax: `(REPLACES &OPTIONAL fnums [*] in[*] from to hide?[T] dir[LR] step[(SKIP)])`

effect: Iterates the PVS command `REPLACE` in the formulas `in`, rewriting with the formulas in `fnums`, respecting the order. The keys `dir` and `hide?` are like in `REPLACE`. Notice that in contrast to `REPLACE`, `true` is the default value of `hide?`. Instead of using `fnums`, rewriting formulas can be addressed via `from` and `to`. The key `step` specifies the command to be executed after all the replaces have taken place.

usage: Given the sequent

```
{-1}  Delta = sq(B) - 4×A×C
{-2}  Delta ≥ 0
{-3}  x = (sqrt(Delta) - B) / (2×A)
  |-------
{1}   A×sq(x) + B×x + C = 0
```

the command `(REPLACES :from -1 :to -3 :in 1)` yields the sequent

```
{-1}  Delta = sq(B) - 4×A×C
{-2}  Delta ≥ 0
  |-------
{1}   A×sq((sqrt(Delta) - B) / (2×A)) +
       B×((sqrt(Delta) - B) / (2×A))
       + C
       = 0
```

Note that since `Delta` does not appear in the original formula {1}, it does not get replaced in the final sequent when the replacements are applied in the order $-1, -2, -3$. In contrast, the command `(REPLACES :from -3 :to -1 :in 1)` yields the sequent

```
{-1}  Delta ≥ 0
  |-------
{1}   A×sq((sqrt(sq(B) - 4×A×C) - B) / (2×A)) +
       B×((sqrt(sq(B) - 4×A×C) - B) / (2×A))
       + C
       = 0
```

The latter sequent is also produced with the command `(REPLACES)`.

4.2. Arithmetic and logic manipulations.

syntax: `(NEG-FORMULA fnum)`

effect: Negates both sides of the relational formula `fnum`.

usage: `(NEG-FOMULA -1)` on sequent

```
{-1}  x < y
```

```
{-2}  a = b
  |-------
{1}   x ≠ a + b ∧ y > a - b
```

produces

```
{-1}  -x > -y
[-2]  a = b
  |-------
[1]   x ≠ a + b ∧ y > a - b
```

syntax: `(ADD-FORMULAS fnum1 fnum2 &OPTIONAL :hide?[T])`
effect: Adds relational formulas `fnum1` and `fnum2`. Hides them when key `hide?` is true.
usage: `(ADD-FOMULAS -1 -2 :hide?  NIL)` on the last sequent yields

```
{-1}  -x + a > -y + b
[-2]  -x > -y
[-3]  a = b
  |-------
[1]   x ≠ a + b ∧ y > a - b
```

syntax: `(WRAP-FORMULA fnum str)`
effect: Wraps both sides of the relational formula `fnum` with the string `str`.
usage: `(WRAP-FORMULA -3 "sq")` on the last sequent yields

```
{-1}  sq(a) = sq(b)
[-2]  -x + a > -y + b
[-3]  -x > -y
[-4]  a = b
  |-------
[1]   x ≠ a + b ∧ y > a - b
```

When proving properties on large arithmetic propositions, involving logical operators, it happens that side conditions have to be proven and then *kept* as hypothesis. The PVS command `SPLIT` is not very effective decomposing this kind of formulas, since it symmetrically splits the propositions. We have implemented an asymmetric split strategy called `SPLASH`.

syntax: `(SPLASH fnum)`
effect: Asymmetrically splits a conjunction `fnum` in the consequent (or a disjunction in the antecedent).
usage: `(SPLASH 1)` on the last sequent yields the two sequents

```
{-1}  x ≠ a + b
[-2]  sq(a) = sq(b)
[-3]  -x + a > -y + b
[-4]  -x > -y
[-5]  a = b
  |-------
{1}   y > a - b
```

```
[-1]  sq(a) = sq(b)
[-2]  -x + a > -y + b
[-3]  -x > -y
[-4]  a = b
  |-------
{1}   x ≠ a + b
```

Note the hypothesis {`-1`} in the first sequent. Now try `(UNDO)`, and compare with `(SPLIT 1)`, where this hypothesis is missing.

5. Conclusion. Hopefully, a decision procedure for the closed field of real number, such as Quantifier Elimination by Cylindrical Algebraic Decomposition [4], will soon be available in PVS [10]. In the mean time, we may have to deal with PVS limitations to handle non-linear arithmetic. Full automation has also its limitations: complex expressions may explode the computational capabilities of the machine. In the other side, proofs by hand are often tedious due to the PVS spartan support of algebraic manipulations.

Based on our experience on verification of avionics systems, we have implemented a package of strategies that combines simple arithmetic manipulation with PVS arithmetic decision procedures. For example, the strategy `FIELD` is a simplification procedure for the field of real numbers that works by removing inverses. The package also provides strategies for cancellation of common terms, negation and addition of formulas, and other common logic and arithmetic manipulation strategies.

The complete package was tuned and tested on several real examples. It has been shown effective in alleviating the proof effort of large and complex formulas. Last, but not least, the set of strategies contained in the package increases the automation power of the theorem prover, without compromising the soundness of the system. Indeed, since the strategies do not make external calls from the system, they are as sound as the original PVS theorem prover.

REFERENCES

[1] B. Barras, S. Boutin, C. Cornes, J. Courant, J. Filliatre, E. Giménez, H. Herbelin, G. Huet, C. Muñoz, C. Murthy, C. Parent, C. Paulin, A. Saïbi, and B. Werner, *The Coq Proof Assistant Reference Manual – Version V6.1*, Tech. Report 0203, INRIA, August 1997. Available at `http://coq.inria.fr/doc-eng.html`.

[2] R. Butler, V. Carreño, G. Dowek, and C. Muñoz, *Formal verification of conflict detection algorithms*, in Proceedings of the 11th Working Conference on Correct Hardware Design and Verification Methods CHARME 2001, vol. 2144 of LNCS, Livingston, Scotland, UK, 2001, pp. 403–417. A long version appears as report NASA/TM-2001-210864.

[3] V. Carreño and C. Muñoz, *Aircraft trajectory modeling and alerting algorithm verification*, in Theorem Proving in Higher Order Logics: 13th International Conference, TPHOLs 2000, J. Harrison and M. Aagaard, eds., vol. 1869 of Lecture Notes in Computer Science, Springer-Verlag, 2000, pp. 90–105. An earlier version appears as report NASA/CR-2000-210097 ICASE No. 2000-16.

[4] G. Collins, *Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition, LNCS 32*, Springer Verlag, 1975.

[5] D. Delahaye and M. Mayero, *Field: une procédure de décision pour les nombres réels en coq.*, tech. report, Unité de recherche INRIA-Rocquencourt, January 2001. Proceedings of JFLA'2001.

Available at `http://pauillac.inria.fr/jfla/2001/`.

[6] G. Dowek, C. Muñoz, and A. Geser, *Tactical conflict detection and resolution in a 3-D airspace*, Tech. Report NASA/CR-2001-210853 ICASE Report No. 2001-7, ICASE-NASA Langley, ICASE Mail Stop 132C, NASA Langley Research Center, Hampton VA 23681-2199, USA, April 2001.

[7] S. Owre, J. M. Rushby, and N. Shankar, *PVS: A Prototype Verification System*, in 11th International Conference on Automated Deduction (CADE), D. Kapur, ed., vol. 607 of Lecture Notes in Artificial Intelligence, Saratoga, NY, June 1992, Springer-Verlag, pp. 748–752.

[8] L. Paulson, *Isabelle: The next 700 theorem provers*, in Logic and Computer Science, P. Odifreddi, ed., Academic Press, 1990, pp. 361–386.

[9] J. Rushby and N. Shankar, *Arithmetic simplification in PVS.* Personal communication, December 2000.

[10] J. Rushby and A. Tiwari, *Formal analysis methods for Mathworks tools.* Personal communication, 2001.

[11] B. D. Vito, *A pvs prover strategy package for common manipulations.* Manuscript. Available at `http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS2-library/pvslib.html`, 2001.