Some Predicate Logic (Spec)

Jonathan Ostroff (EECS)

October 23, 2018

Revisions

Date	Revision	Description
12 September 2017	1.0	Initial notes for predicate logic

Contents

List of Figures

On your own, please do all the parts that are highlighted.

1. Square Root of Two is not Rational

1.1. Some preliminary Lemmas

Mathematicians normally write theorems in a semi-formal way. For example:

Theorem 1 Quotient Remainder Theorem: Given any integer n and positive integer d there exist unique integers q and r so that $n = dq + r$ and $0 \leq r < d$.

Thus for $n = -54$ and $d = 4$, the quotient $q = -14$ (i.e. $n \div d$) and the remainder $r = 2$ (i.e. $n \mod d$).

Interestingly, languages such as C and Java give values that satisfy the remainder theorem for nonnegative integers, but not for negative integers n.[1]

Likewise, definitions are written informally, for example:

Definition 1 An integer n is *even* if $n = 2a$ for some integer a.

Formally, we might write: $even(n) \mathrel{\widehat{=}} (\exists a \in \mathbb{Z} : n = 2a)$.

Definition 2 An integer is odd if $n = 2a + 1$ for some integer $a \in \mathbb{Z}$.
Thus: Thus: $odd(n) \mathrel{\widehat{=}} (\exists a \in \mathbb{Z} : n = 2a + 1)$.

You should be able to prove as a theorem of arithmetic that every integer is either even or odd. How would you prove that? Here is another Lemma.

Lemma 1 $\neg even(n) \equiv odd(n)$, for all integers n.
Also, we can then derive: $\neg odd(n) \equiv even(n)$.

Note that $n \in \mathbb{Z}$ is a free-variable in this Lemma, and thus what must be proved is really: $\forall n \in \mathbb{Z} : \neg even(n) \equiv odd(n)$

Proof: Can you do this proof equationally and then in PVS?. Quite a bit of help is provided below and in the Appendix.

[1] Java: "int r = n % d; int q = n / d" yields quotient = -13 and remainder = -2. Python: "q = n // d; r = n % d" yields the mathematical answer of quotient = -14 and remainder = 2. Challenge: write a Function Table to specify how Java behaves and how Python behaves. Allow n and d to be any integers. What about when n and d are reals?

1.2. Equational Logic

See [2] (the textxbook for your required course MATH109) and [1] for equational logic. A list of useful theorems may be found in [1]. For example, GS3.61: $p \Rightarrow q \equiv \neg q \Rightarrow \neg p$.

A complete list is pf theorems available online at `http://www.eecs.yorku.ca/course_archive/2017-18/F/4312/GS-Theorems.pdf`. See Appendix A for more on Equational Logic.

Types (in a typed logic) are as follows:

Name	Symbol	Type (set of values)
integer	$\mathbb{Z}$	integers: $\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots$
nat	$\mathbb{N}$	natural numbers: $0, 1, 2, \ldots$
positive	$\mathbb{Z}^+$	positive integers: $1, 2, 3, \ldots$
negative	$\mathbb{Z}^-$	negative integers: $-1, -2, -3, \ldots$
rational	$\mathbb{Q}$	rational numbers: i/j, $i, j \in \mathbb{Z}$, $j \neq 0$
reals	$\mathbb{R}$	real numbers
positive reals	$\mathbb{R}^+$	positive real numbers
bool	$\mathbb{B}$	booleans: *true*, *false*

1.3. Doing these proofs in PVS

See Appendix C where we provide substantial help to do the theorems in this paper. Here are some proofs to try in PVS:

```
conj4: CONJECTURE (FORALL (i:int): NOT (even?(i) AND odd?(i)))

conj6: CONJECTURE (FORALL (i:int): even?(i) = (NOT odd?(i)))
```

1.4. A Lemma about even numbers

Consider the following proposition and proof taken from a mathematical textbook.

This is an *informal* proof. Is it a good proof leaving no room for doubt? Is it clear what facts are being used at each step, and in fact what each step is?

Equational Proofs

Equational proofs provide more clarity by working in relatively small steps and providing the justification for each step, usually with the help of the Leibniz or monotonicity proof rule. Try to prove this Lemma on your own in equational logic, before proceeding.

Proposition 4.6.4

For all integers n, if n^2 is even then n is even.

Proof (by contraposition):

Suppose n is any odd integer. *[We must show that n^2 is odd.]* By definition of odd, $n = 2k + 1$ for some integer k. By substitution and algebra,

$$n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1.$$

But $2k^2 + 2k$ is an integer because products and sums of integers are integers. So $n^2 = 2 \cdot$(an integer) $+ 1$, and thus, by definition of odd, n^2 is odd *[as was to be shown]*.

We start with a small Lemma about $odd(n)$:

Lemma 2 For all integers n, if n is odd then n^2 is odd.
Thus: $\forall n \in \mathbb{Z} : odd(n) \Rightarrow odd(n^2)$.

Proof of Lemma 2

In the proof below, GS9.30 is a meta-rule for existential elimination:

$$\frac{H, P[x := w] \vdash G}{H, (\exists x : P) \vdash G}$$ Rule ∃Hyp, where w is a fresh constant

Let n be any arbitrary integer n (In PVS this would be done by skolemization). Then:

$odd(n)$
$=$ {Defn. 2 for odd}
$\exists a \in \mathbb{Z} : n = 2a + 1$
$\Rightarrow$MR {Meta-theorem GS9.30: Existential elimination for some new witness k}
$n = 2k + 1$
$=$ {Algebra: squaring both sides}
$n^2 = (2k + 1)^2$
$=$ {Algebra: $(2k + 1)^2 = (4k^2 + 4k + 1) = 2(2k^2 + 2k) + 1$}
$n^2 = 2(2k^2 + 2k) + 1$
$=$ {for some integer $j = 2k^2 + 2$; products and sums of integers are integers}
$n^2 = 2j + 1$
$\Rightarrow$ {existential introduction GS9.28: $P[x := n] \Rightarrow (\exists n : P(n))$}
$\exists j \in \mathbb{Z} : n^2 = 2j + 1$
$\Rightarrow$ {Defn. 2 for odd}
$odd(n^2)$

Since n was an arbitrary integer: $\forall n \in \mathbb{Z} : odd(n) \Rightarrow odd(n^2)$ is a theorem. **QED**.

Note: $\Rightarrow$MR indicates that we need to temporarily step outside an equational proof to use the GS9.30 meta-rule.

In Fig. 1, the Skolem/Skeep rules require a new candidate c variable that does not occur free in H. As usual, $P[x := c]$ stands for the predicate similar to P but with free occurrence of x replaced by c, where capture does not occur.

Lemma 3 For all integers n, if n^2 is even then n is even.
Thus: $\forall n \in \mathbb{Z} : even(n^2) \Rightarrow even(n)$.

Proof

The proof will be by contrapositive GS3.61.

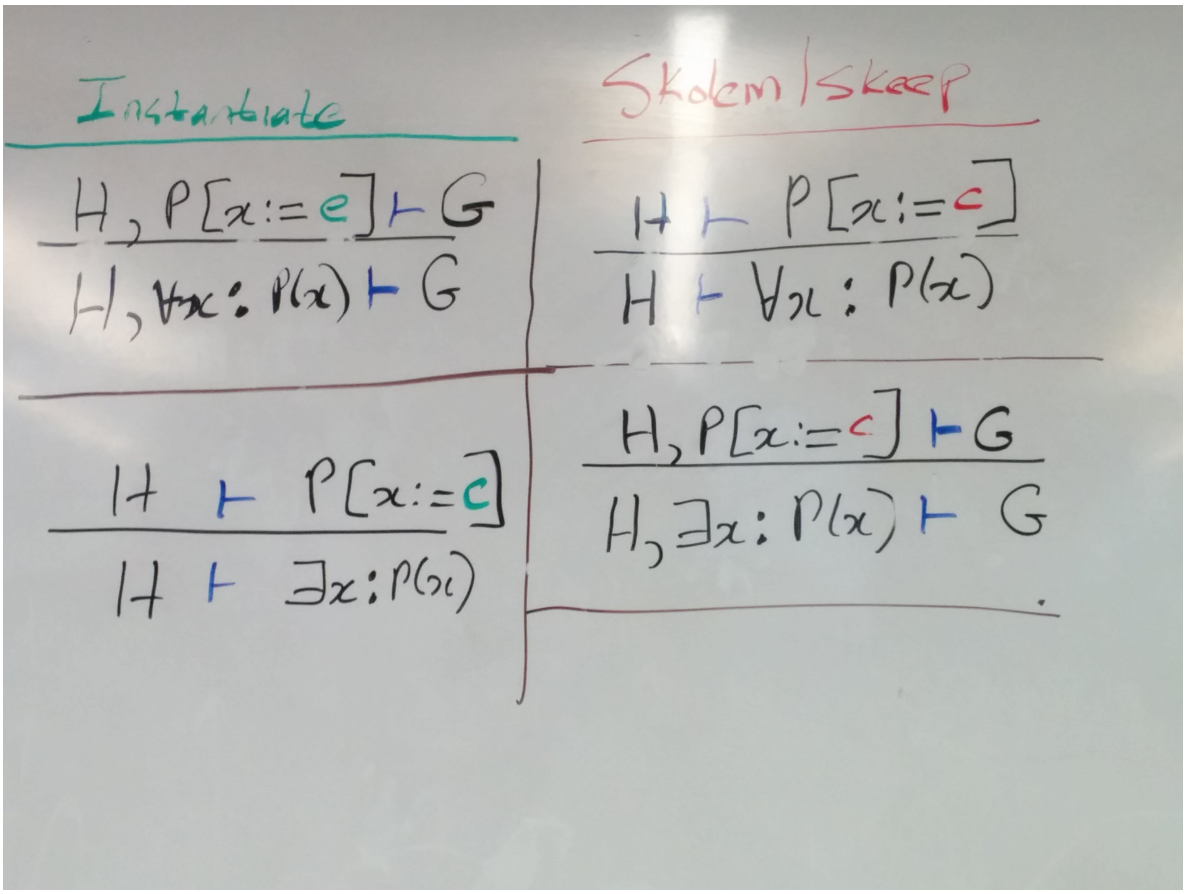

Figure 1: PVS Sequent Calculus Meta-Rules for Predicate Logic

$$
\begin{aligned}
& odd(n) \Rightarrow odd(n^2) && \text{-- Lemma 2} \\
= & \quad \{\text{GS3.61: } p \Rightarrow q \ \equiv \ \neg q \Rightarrow \neg p\} \\
& \neg odd(n^2) \Rightarrow \neg odd(n) \\
= & \quad \{\text{Lemma 1: } \neg odd(x) \equiv even(x)\} \\
& even(n^2) \Rightarrow even(n)
\end{aligned}
$$

Since n was an arbitrary integer: $\forall n \in \mathbb{Z} : even(n^2) \Rightarrow even(n)$ is a theorem. **QED**.

Note that the proof shows that $even(n^2) \Rightarrow even(n)$ is equivalent to a Lemma that has already been proved. Thus $even(n^2) \Rightarrow even(n)$ has also been proved.

1.5. Square Root of Two

The following is a textbook example of a proof that the $\sqrt{2}$ is irrational.

Theorem 2 $\sqrt{2}$ is irrational.

Suppose $\sqrt{2}$ is rational.
Then $\sqrt{2} = \frac{m}{n}$ where $m, n \in \mathbb{N}$ and $\gcd(m, n) = 1$.
Squaring, $2 = m^2/n^2 \implies 2n^2 = m^2 \implies m = 2k$ for some $k \in \mathbb{N}$ (m is even).
Then $2n^2 = 4k^2 \implies n^2 = 2k^2$, so n must be even.
But now we see that 2 divides both m and n, which contradicts the initial assumption that $\gcd(m, n) = 1$.
Hence $\sqrt{2}$ is irrational.

–

Figure 2: Traditional Proof that $\sqrt{2}$ is irrational

Do you understand the proof? Can you reproduce it?

An equational proof might help you understand it better. Try it equationally, and then try it in PVS (see Appendix).

1.5.1. Proof that $\sqrt{2}$ is irrational in equational logic

Proof by contradiction: Assume not, i.e. assume $\sqrt{2}$ is rational. Then there are integers m and n such that $\sqrt{2} = m/n$. Let this fraction be fully reduced (i.e. divide m and n by common factors if necessary). If they are both even, the fraction could be further reduced by dividing by 2. Thus, under our assumption, m and n cannot both be even, i.e.

$$\begin{aligned} \sqrt{2} \text{ is rational } &\Rightarrow (\sqrt{2} = m/n) \wedge gcd(n, m) = 1 \\ &\Rightarrow \neg(even.m \wedge even.n) \end{aligned} \tag{1}$$

We will now show that:

$$\begin{aligned} \sqrt{2} \text{ is rational } &\Rightarrow (\sqrt{2} = m/n) \wedge gcd(n, m) = 1 \\ &\Rightarrow even.m \wedge even.n \end{aligned} \tag{2}$$

Do the rest of the proof on you own. I.e. present the proof of of Theorem (2) equationally.

1.5.2. Proving $\sqrt{2}$ is irrational in PVS

The first question is how to formalize the the theorem in PVS?

In Appendix C we use two of the NASA libraries (*sqrt* and *gcd*) to formalize the theorem as follows:

```
IMPORTING reals@sqrt
IMPORTING ints@gcd

PosRational? : PRED[real] =
    { t : real | EXISTS(m, n:posnat) : t = m/n AND gcd(n,m) = 1 }

sqrt2_non_rational: THEOREM NOT PosRational?(sqrt(2))
```

We first prove the following Lemmas, and then use the Lemmas to prove the theore,

```
sqrt2_parta: LEMMA
   (FORALL (m,n:posnat):
   sqrt(2)= m/n AND gcd(n, m)=1 => NOT (even?(m) AND even?(n)))


sqrt2_partb: LEMMA
   (FORALL (m,n:posnat):
   sqrt(2)=m/n AND gcd(n, m)=1 =>  even?(m) AND even?(n))
```

Now do the proofs, using the guidance provided in the Appendix, on your own

You can search the PVS libraries for helpful definition and theorems using the Hypatheon (shown in Fig. 3) tool (invoked from the command line as `hypatheon`).

See Appendix B for how to use PVS to compute recursive functions and output computations to the screen.

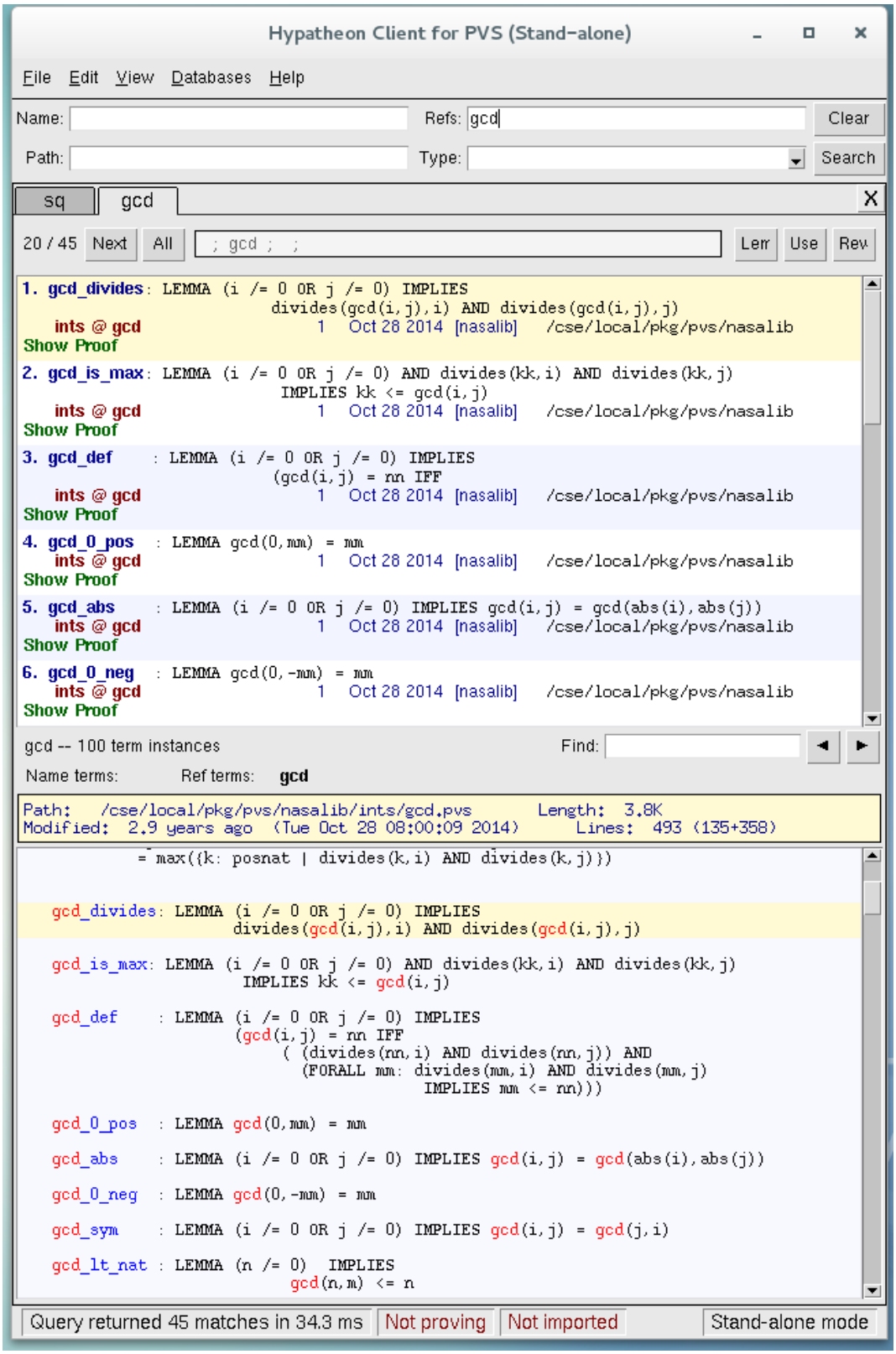

Figure 3: Hypatheon

References

[1] David Gries and Fred B. Schneider. *A Logical Approach to Discrete Math.* Springer Verlag, 1993.

[2] George Tourlakis. *Mathematical Logic.* Wiley, 2008.

A. Equational Logic

See [2] and [1] for equational logic.

A.1. Some Rules of Equational Logic

$$\frac{A, A \equiv B}{B}\ \text{Equanimity} \qquad \frac{A \equiv B}{C[\mathbf{p} := A] \equiv C[\mathbf{p} := B]}\ \text{Leibniz; no illegal capture}$$

$C[\mathbf{p} := A]$ is the formula similar to C except that every *free* occurrence of $\mathbf{p}$ in C is replaced by A, provided that there is no illegal capture.

For example, suppose C is $(\forall x : \mathbf{p})$ then in $C[\mathbf{p} := (y = 2x)]$, the variable x in the predicate $(y = 2x)$ will be captured by the quantifier $\forall x$; thus, $C[\mathbf{p} := (y = 2x)]$ is undefined.

$$\frac{H, A \vdash B}{H \vdash A \Rightarrow B}\ \text{Deduction Theorem}$$

For predicate logic we have, provided H has no free occurrence of x:

$$\frac{H \vdash A \equiv B}{H \vdash (\forall x : C[\mathbf{p} := A]) \ \equiv \ (\forall x : C[\mathbf{p} := B])}\ 8.12\text{a; p169}$$

$$\frac{H, D \vdash A \equiv B}{H \vdash (\forall x : D \Rightarrow C[\mathbf{p} := A]) \ \equiv \ (\forall x : D \Rightarrow C[\mathbf{p} := B])}\ 8.12\text{b; p169}$$

A.2. Monotinicity (Weakening)

Provided the position of $\mathbf{p}$ in A is monotonic, then:[2]

$$\frac{A \Rightarrow B}{A[\mathbf{p} := A] \Rightarrow A[\mathbf{p} := B]}\ \text{Monotonicity; Weakening}$$

A function f is monotonic in its argument if $x \Rightarrow y$ implies $f.x \Rightarrow f.y$ (for all x,y). The propositional operators $\vee$ and $\wedge$ are monotonic in their operands.

[2] http://www.eecs.yorku.ca/course_archive/2017-18/F/4312/GS-Monotinicity.pdf

$$
\begin{array}{ll}
 & < \forall x : A \wedge B > \\
\Rightarrow & \quad \{\text{Weakening } A \Rightarrow A \vee B\} \\
 & < \forall x : (A \vee B) \wedge B >
\end{array}
$$

A.3. A proof in equational logic

(1) $p \;\Rightarrow\; n \equiv (m \wedge \neg t) \vee (\neg m \wedge t))$
(2) $n \wedge \neg t \;\Rightarrow\; o$
(3) $p \wedge m \wedge \neg t \;\Rightarrow\; o$
Assume p, thus $n \equiv (m \wedge \neg t) \vee (\neg m \wedge t)$

$$
\begin{array}{ll}
 & n \wedge \neg t \;\Rightarrow\; o \qquad\qquad \text{-- hypothesis (2)} \\
= & \quad \{\text{Assumption } p, \text{ Leibniz}\} \\
 & ((m \wedge \neg t) \vee (\neg m \wedge t)) \wedge \neg t \;\Rightarrow\; o \\
= & \quad \{\text{GS-3.46 Distributivity of } \wedge\} \\
 & (m \wedge \neg t \wedge \neg t) \vee (\neg m \wedge t \wedge \neg t) \;\Rightarrow\; o \\
= & \quad \{\text{GS-3.42 Contradiction: } t \wedge \neg t \;\equiv\; \bot\text{; GS-3.40 Zero of } \wedge\} \\
 & (m \wedge \neg t \wedge \neg t) \vee (\bot) \;\Rightarrow\; o \\
= & \quad \{\text{ GS-3.30 Identity of } \vee\text{; GS-3.38: } \neg t \wedge \neg t \;\equiv\; \neg t\} \\
 & (m \wedge \neg t) \;\Rightarrow\; o
\end{array}
$$

We assumed p, so by the deduction theorem: $p \wedge m \wedge \neg t \;\Rightarrow\; o$

B. PVSIO can compute via recursive functions

A measure must be provided and a TCC proved to ensure termination.

```
ground  : THEORY
BEGIN

% Also defined in the NASA reals library
sqrt_newton(a:nnreal,n:nat): RECURSIVE posreal =
  IF n=0 THEN a+1
  ELSE let r=sqrt_newton(a,n-1) IN
    (1/2)*(r+a/r)
  ENDIF
  MEASURE n+1

%M-x pvsio
% <PVSio> (sqrt_newton(2,1));
% 11/6
% <PVSio> (sqrt_newton(2,3));
% 72097/50952
%println(sqrt_newton(2,10));
%<PVSio> println(sqrt_newton(2,10));
%1.4142135

%Has been shown to satisfy
% sqrt(a) < sqrt_newton(a, n)
% sqrt_newton(a, n+1) < sqrt_newton(a,n)
% As n --> infinity, sqrt_newton(a,n) converges to sqrt(q)

test: CONJECTURE
      2 < sqrt_newton(2, 10) * sqrt_newton(2, 10)
      AND
      sqrt_newton(2, 10) * sqrt_newton(2, 10) < 2.0000000000000001

%|- test : PROOF
%|- (eval-formula 1)
%|- QED
% PVSio safely enables the ground evaluator in the theorem
% prover.
END ground
```

C. PVS theory for $\sqrt{2}$ irrational

```
sqrt2: THEORY
  % Proof that sqrt(2) is not rational
  % Includes axiom1 that should still be proved
  % See detailed instuctions at the end of this file.
BEGIN
  IMPORTING reals@sqrt
  IMPORTING ints@gcd

  % From reals@sqrt
  %nnx: VAR nonneg_real
  %sqrt(nnx): {nnz : nnreal | nnz*nnz = nnx}
  sqrt4: LEMMA  sqrt(4) = 2

%|- sqrt4 : PROOF
%|- (assert)
%|- QED

   % From ints@gcd
   % gcd(i:int,j: {jj:int| i=0 => jj /= 0}):
   %       {k: posnat | divides(k,i) AND divides(k,j)}
   %          = max({k: posnat | divides(k,i) AND divides(k,j)})

  % From ints@gcd
  gcd_conjecture: CONJECTURE gcd(12,18) = 6

%|- gcd_conjecture : PROOF
%|- (then (typepred "compute_gcd(12,18)") (assert) (expand "compute_gcd")
%|-  (expand "gcd") (assert) (expand "divides") (ground) (grind))
%|- QED

  %both_sides_times1: LEMMA (x * n0z = y * n0z) IFF x = y
  %both_sides_times2: LEMMA (n0z * x = n0z * y) IFF x = y
  %divides(n,m:int): bool = (EXISTS (x:int): n*x = m)

  divides_conjecture: CONJECTURE divides(3,12)

%|- divides_conjecture : PROOF
%|- (then (expand "divides") (inst 1 4) (assert))
%|- QED
```

```
  % http://pvs.eecs.yorku.ca/prelude
  % integers: THEORY in the prelude
  % even?(i): bool = EXISTS j: i = j * 2
  % odd?(i): bool  = EXISTS j: i = j * 2 + 1
  % even_int: NONEMPTY_TYPE = (even?) CONTAINING 0
  % odd_int:  NONEMPTY_TYPE = (odd?)  CONTAINING 1

   axiom1: AXIOM (FORALL (i:int): even?(i) OR odd?(i))
   % Prove this with induction

  conj1: CONJECTURE even?(4)

  conj2: CONJECTURE odd?(3*5)
  conj3: CONJECTURE (FORALL (i,j:int): 2*i /= 1 + 2*j)

%|- conj3 : PROOF
%|- (then (skeep) (assert))
%|- QED

  conj4: CONJECTURE (FORALL (i:int): NOT (even?(i) AND odd?(i)))

%|- conj4 : PROOF
%|- (then (skeep) (expand "even?") (expand "odd?") (skeep -1) (skeep -2)
%|-  (replace -1) (hide -1) (lemma "conj3" ("i" "j" "j" "j!1")) (assert))
%|- QED

  %conj5: CONJECTURE (FORALL (i:int): even?(i) => NOT odd?(i))

%|- conj5 : PROOF
%|- (then (skeep) (expand "even?") (expand "odd?") (skeep -1) (skeep -2)
%|-  (assert))
%|- QED

  conj5: CONJECTURE (FORALL (i:int): (NOT odd?(i)) = even?(i))

%|- conj5 : PROOF
%|- (then (skeep) (iff)
%|-  (spread (split)
%|-   ((then (flatten) (lemma "axiom1") (inst -1 i) (prop))
%|-    (then (flatten) (grind)))))
```

```
%|- QED

  conj6: CONJECTURE (FORALL (i:int): even?(i) = (NOT odd?(i)))

%|- conj6 : PROOF
%|- (then (skeep) (iff)
%|-  (spread (split)
%|-   ((then (flatten) (lemma "axiom1") (inst -1 i)
%|-     (spread (prop) ((grind) (grind))))
%|-    (then (flatten) (lemma "axiom1") (inst -1 i) (prop)))))
%|- QED

  conj7: CONJECTURE (FORALL (x,y:int): (x=y) => (x*x = y*y))

%|- conj7 : PROOF
%|- (grind)
%|- QED

  conj8: CONJECTURE (FORALL (n:int): odd?(n) => odd?(n*n))

%|- conj8 : PROOF
%|- (then (skeep) (expand "odd?" -1) (skolem -1 "k") (lemma "conj7")
%|-  (inst -1 "n" "1+2*k") (simplify) (prop) (expand "odd?")
%|-  (inst 1 "2*k*k + 2*k") (grind))
%|- QED

  conj9: CONJECTURE (FORALL (n:int): even?(n*n) => even?(n))

%|- conj9 : PROOF
%|- (then (skeep) (lemma "conj8") (inst -1 n) (lemma "conj6")
%|-  (inst-cp -1 "n*n") (replace -2) (inst -1 n) (replace -1) (prop))
%|- QED

   PosRational? : PRED[real] =
    { t : real | EXISTS(m, n:posnat) : t = m/n AND gcd(n,m) = 1 }

 % The following TCC is produced for PosRational?(5)
  % FORALL (t: real, n, m: posnat): t = n / m IMPLIES m = 0 => n /= 0;
  % I am not sure why we must do (expand "compute_gcd") twice?
  conj10: CONJECTURE
    PosRational?(5)
```

```
%|- conj10 : PROOF
%|- (then (expand "PosRational?") (inst 1 "5" "1")
%|-  (typepred "compute_gcd(1,5)") (expand "compute_gcd")
%|-  (expand "compute_gcd") (grind))
%|- QED

   % From theory gcd: gcd_times: LEMMA gcd(p * l, q * l) = gcd(p, q) * l
   % Thus: gcd(i * 2, j * 2) = gcd(i, j) * 2
  sqrt2_parta: LEMMA
   (FORALL (m,n:posnat):  sqrt(2) = m / n AND gcd(n, m) = 1 =>  NOT (even?(m) AND
  %% TODO: You must do this proof using previous lemmas and axiom
  %% See Proof-sqrt2_parta.txt for detailed help

  sqrt2_partb: LEMMA
   (FORALL (m,n:posnat):  sqrt(2) = m / n AND gcd(n, m) = 1 =>  even?(m) AND even
  %% TODO: You must do this proof using previous lemmas and axiom
  %% See Proof-sqrt2_partb.txt for detailed help

   sqrt2_non_rational: THEOREM NOT PosRational?(sqrt(2))
   %% TODO: You must do this proof using previous lemmas and axiom

END sqrt2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% When you start, you will see the following.
%%%% Note that if you do C-c C-t, i.e. typecheck you will see that there are
%%%% some unproved type correctness conditions.
%%%% Ensure that you understand them. They can be proved automatically
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 % Proof summary for theory sqrt2
 %    sqrt4.................................untried              [Untried]( n/a s)
 %    gcd_conjecture........................untried              [Untried]( n/a s)
 %    divides_conjecture....................untried              [Untried]( n/a s)
 %    conj1.................................untried              [Untried]( n/a s)
 %    conj2.................................untried              [Untried]( n/a s)
 %    conj3.................................untried              [Untried]( n/a s)
 %    conj4.................................untried              [Untried]( n/a s)
 %    conj5.................................untried              [Untried]( n/a s)
 %    conj6.................................untried              [Untried]( n/a s)
 %    conj7.................................untried              [Untried]( n/a s)
```

```
 %    conj8.................................untried              [Untried]( n/a s)
 %    conj9.................................untried              [Untried]( n/a s)
 %    PosRational?_TCC1.....................proved - complete   [shostak](0.06 s)
 %    conj10................................untried              [Untried]( n/a s)
 %    sqrt2_parta_TCC1......................proved - complete   [shostak](0.00 s)
 %    sqrt2_parta...........................untried              [Untried]( n/a s)
 %    sqrt2_partb...........................untried              [Untried]( n/a s)
 %    sqrt2_non_rational....................untried              [Untried]( n/a s)
 %    Theory totals: 18 formulas, 2 attempted, 2 succeeded (0.06 s)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% some unproved type correctness conditions.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Subtype TCC generated (at line 114, column 57) for  m
%     % expected type  {jj: int | n = 0 => jj /= 0}
%   % proved - complete
% PosRational?_TCC1: OBLIGATION
%   FORALL (t: real, m, n: posnat): t = m / n IMPLIES n = 0 => m /= 0;

% % Subtype TCC generated (at line 131, column 53) for  m
%     % expected type  {jj: int | n = 0 => jj /= 0}
%   % proved - complete
% sqrt2_parta_TCC1: OBLIGATION
%   FORALL (m, n: posnat): sqrt(2) = m / n IMPLIES n = 0 => m /= 0;

% The subtype TCC (at line 136, column 53) in decl sqrt2_partb for
m
    % expected type  {jj: int | n = 0 => jj /= 0}
  % is subsumed by sqrt2_parta_TCC1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Below is what you should see when you are done.
%%%% Full Proveit scripts are provided to help you
%%%% See also where its says TODO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Proof summary for theory sqrt2
%     sqrt4.................................proved - complete   [shostak](0.01 s)
%     gcd_conjecture........................proved - complete   [shostak](0.19 s)
%     divides_conjecture....................proved - complete   [shostak](0.02 s)
%     conj1.................................proved - complete   [shostak](0.01 s)
```

```
%     conj2.................................proved - complete   [shostak](0.02 s)
%     conj3.................................proved - complete   [shostak](0.04 s)
%     conj4.................................proved - complete   [shostak](0.05 s)
%     conj5.................................proved - complete   [shostak](0.06 s)
%     conj6.................................proved - complete   [shostak](0.10 s)
%     conj7.................................proved - complete   [shostak](0.01 s)
%     conj8.................................proved - complete   [shostak](0.12 s)
%     conj9.................................proved - complete   [shostak](0.03 s)
%     PosRational?_TCC1.....................proved - complete   [shostak](0.00 s)
%     conj10................................proved - complete   [shostak](0.08 s)
%     sqrt2_parta_TCC1......................proved - complete   [shostak](0.00 s)
%     sqrt2_parta...........................proved - complete   [shostak](0.10 s)
%     sqrt2_partb...........................proved - complete   [shostak](0.20 s)
%     sqrt2_non_rational....................proved - complete   [shosftak](0.03 s
%     Theory totals: 18 formulas, 18 attempted, 18 succeeded (1.08 s)

% compute_gcd(i:int,j:{jj:int| i=0 => jj /= 0}): RECURSIVE {kj: posnat | kj = gcd
%      IF    i<0 THEN compute_gcd(-i,j)
%      ELSIF j<0 THEN compute_gcd(i,-j)
%      ELSIF i=0 THEN j
%      ELSIF j=0 THEN i
%      ELSIF i<j THEN compute_gcd(j,i)
%      ELSE
%        (LET rem = mod(i,j) IN
%          IF rem = 0 THEN j ELSE compute_gcd(j,rem) ENDIF)
%      ENDIF
%        MEASURE (IF i<0 AND j<0 THEN -i-j+3
%                        ELSIF i<0 THEN -i+j+2
%                ELSIF j<0 THEN i-j+2
%                ELSIF i<j THEN i+j+1
%                ELSE i+j ENDIF)
```