Square Root Proof with Coq

Yves Bertot, Loïc Pottier, Laurence Rideau, Paul Zimmermann

Lemme and PolKA projects, INRIA Sophia and Lorraine

Our (initial) objective

Prove the GMP square root implementation, starting from the hand-written proof at
`http://www.loria.fr/~zimmerma/papers/proof-div-sqrt.ps.gz`

Tools: Coq V6.3.1 (proof assistant) and Pcoq V1.1 (user interface)

Sketch of the implementation

$$0 \le N \quad \longrightarrow \quad N = S^2 + R \quad \text{with} \quad 0 \le R \le 2S$$

Four functions (word base $B = 2^{32}$ or 2^{64}):

`mpn_sqrtrem1`: $B/4 \le N < B$

`mpn_sqrtrem2`: $B^2/4 \le N < B^2$

`mpn_dc_sqrtrem`: $B^{2n}/4 \le N < B^{2n}$

`mpn_sqrtrem_main`: top-level function

The `mpn_dc_sqrtrem` function

Input: $\frac{B^{2n}}{4} \leq N < B^{2n}$

Output: (S, R) such that $N = S^2 + R$ with $0 \leq R \leq 2S$

if $n = 1$ **then** return(`mpn_sqrtrem2`(N))
$h = \lceil n/2 \rceil, l = \lfloor n/2 \rfloor$
$N = N' B^{2l} + N''$ with $0 \leq N'' < B^{2l}, \quad B^{2h}/4 \leq N' < B^{2h}$
$(S', R') =$ `mpn_dc_sqrtrem`(N')
$N'' = N_1 B^l + N_0$ with $0 \leq N_1, N_0 < B^l$
$(Q, R'') =$ `DivRem`$(R' B^l + N_1, 2S')$
$S_1 = S' B^l + Q$
$R_1 = R'' B^l + N_0 - Q^2$
if $R_1 < 0$ **then** $R = R_1 + 2S_1 - 1, S = S_1 - 1$
else $R = R_1, S = S_1$

What we've done

1. Proof for one induction step

   (generalized B^h to H, B^l to L, B^n to HL)

2. Proof for the whole recursive function

   (includes proof of termination)

One induction step

$P_1(N,H,S,R) \iff N = S^2 + R$

$P_2(N,H,S,R) \iff R \geq 0$

$P_3(N,H,S,R) \iff R \leq 2S$

$P_4(N,H,S,R) \iff H/2 \leq S$ i.e. $H \leq 2S$

$P_5(N,H,S,R) \iff S < H$

H1: $0 < L \leq H$

H2: $(HL)^2/4 \leq N < (HL)^2$

H3: $P_i(N', H, S', R')$ for $1 \leq i \leq 5$

To be proven: $P_i(N, HL, S, R)$ for $1 \leq i \leq 5$

Auxiliary lemmas

We had to prove (some with difficulty) auxiliary lemmas which were not available in the default tactics (ZArith) over the integers:

$$\forall n, m, p \in \mathbb{Z}, n \geq m \Rightarrow m \geq p \Rightarrow n \geq p$$

$$\forall a, b \in \mathbb{Z}, a > 0 \Rightarrow b > 0 \Rightarrow ab > 0$$

$$\forall a, b, c \in \mathbb{Z}, c > 0 \Rightarrow ac \geq bc \Rightarrow a \geq b$$

$$\forall a, b, c \in \mathbb{Z}, c > 0 \Rightarrow ac > bc \Rightarrow a > b$$

$$\forall a, b, c, d \in \mathbb{Z}, a < b \Rightarrow c \leq d \Rightarrow ac < bd$$

$$A_2 : \forall a, b, c, d \in \mathbb{Z}, a \geq c \Rightarrow b \geq d \Rightarrow c \geq 0 \Rightarrow d \geq 0 \Rightarrow ab \geq cd$$

$$\forall x, y \in Z, x \geq 0 \Rightarrow y \geq 0 \Rightarrow xx > yy \Rightarrow x > y$$

```
Lemma L7: (x, y : Z) (Zge x ZERO) -> (Zge y ZERO) ->
                     (Zgt (Zmult x x) (Zmult y y)) -> (Zgt x y).
Intros x y H0 H1 H2.
Case (Z_lt_dec y x).
Intros; Omega.
Intros H3; Cut (Zge y x).
Intros H.
Elim Zgt_not_le with 1 := H2.
Apply Zge_le.
Apply A2; Auto.
Omega.
Qed.
```

Results (1)

Starting from zero knowledge about Coq and its nice interface Pcoq, we were able in one week (with a great assistance from the experts of Lemme) to get a proof of one induction step.

The proof is about 600 lines long (including auxiliary lemmas, wrt 2 pages for the hand-written proof), and takes about 100 seconds to be checked by Coq (about 100 minutes for the hand-written proof?):

```
zimmerma@ecrouves aoc]$ wc -l proof_sqrt.v
    617 proof_sqrt.v
[zimmerma@ecrouves aoc]$ time coqc -image coq-contrib.out proof_sqrt.v
Welcome to Coq V6.3.1 (December 1999)
...
97.700u 0.490s 1:39.10 99.0%    0+0k 0+0io 1108pf+0w
```

Example: proof of P_1

```
Lemma T1: (SqrtremProp1 N (Zmult H L) S2 R2).
Red.
Case (Z_lt_dec R1 ZERO).
Intros z; Try Assumption.
Rewrite (H7 z). Rewrite (H8 z). Clear z.
Unfold Zsquare Ztwo Zone. Ring.
Elim N'N''def; Intros H' Dummy; Clear Dummy; Rewrite H'.
Elim N1N0def; Intros H'0 Dummy; Clear Dummy; Rewrite H'0.
Elim H11.
Intros H1 H2; Rewrite H1.
Unfold S1; Unfold R1.
Replace N1 with (Zplus (Zmult Q (Zmult Ztwo S')) (Zminus R'' (Zmult R' L)))
Unfold Zsquare Ztwo Zone. Ring.
Replace N1 with (Zminus (Zplus (Zmult R' L) N1) (Zmult R' L)).
```

```
Elim QR''def; Intros H'1 Dummy; Rewrite H'1; Clear Dummy.
Unfold Zsquare Ztwo Zone. Ring. Ring.
Intros z.
Rewrite (H9 z). Rewrite (H10 z). Clear z.
Elim N'N''def; Intros H' Dummy; Clear Dummy; Rewrite H'.
Elim N1N0def; Intros H'0 Dummy; Clear Dummy; Rewrite H'0.
Elim H11; Intros H'1 Dummy; Clear Dummy; Rewrite H'1.
Unfold S1; Unfold R1.
Replace N1 with (Zplus (Zmult Q (Zmult Ztwo S')) (Zminus R'' (Zmult R' L)))
Unfold Zsquare Ztwo Zone.
 Ring.
Replace N1 with (Zminus (Zplus (Zmult R' L) N1) (Zmult R' L)).
Elim QR''def; Intros H'2 Dummy; Clear Dummy; Rewrite H'2.
Unfold Zsquare Ztwo Zone.
 Ring.
 Ring.
Qed.
```

Results (2)

The 2nd part of the proof was done by Yves Bertot and Laurence Rideau:

```
[zimmerma@ecrouves aoc]$ wc -l fun.v
    571 fun.v
[zimmerma@ecrouves aoc]$ time coqc -image coq-contrib.out fun.v
Welcome to Coq V6.3.1 (December 1999)
...
sqrtrem is defined
23.150u 0.330s 0:23.67 99.1%    0+0k 0+0io 1234pf+0w
```

Some non-trivial lemmas

$$\forall z \in \mathbb{Z}, \forall h, l \in \mathbb{N}, \quad z^{hl} = z^h z^l$$

$$\forall h \in \mathbb{N}, h > 0 \Rightarrow \exists v \in \mathbb{Z}, 4v = b^{2h} \quad (b \text{ even and positive})$$

$$\forall P : (\mathbb{N} \to \text{bool}), P(0) \Rightarrow P(1) \Rightarrow (\forall n \in \mathbb{N}, P(n) \Rightarrow P(n{+}2)) \Rightarrow (\forall n \in \mathbb{N}, P(n))$$

$$\forall n \in N, 2\lceil n/2 \rceil = n \quad \text{or} \quad 2\lceil n/2 \rceil = n + 1$$

$$\forall n \in N, 2\lfloor n/2 \rfloor \leq n$$

$$\forall n \in N, \lceil n/2 \rceil + \lfloor n/2 \rfloor = n$$

```
Fixpoint ceildiv2' [n : nat] : nat :=
 Cases n of   O => O
            | (S O) => (S O)
            | (S (S n)) => (S (ceildiv2' n))
 end.

Definition sqrt_F: (h : nat) (f : (h' : nat) (lt h' h) -> (sqrt_F_type h'))
                    (sqrt_F_type h).
Unfold 2 sqrt_F_type.
Intros h sqrtrem n Hnorm.
Cut (Zge n ZERO). Intros NgeZERO.
Refine (Case (le_lt_dec h (1)) of [Hyp : (le h (1))]
             (Case (sqrtrembase n)
                   of [p : Z * Z]
                      [Hbase : ?]
                   (exist ? ? p ?)
...
```

```
Definition sqrtrem : (h : nat) (sqrt_F_type h) :=
   (well_founded_induction
      nat
      lt
      lt_wf
      [x : nat]  (sqrt_F_type x)
      sqrt_F).
```

Conclusions

Some of the difficulties with Coq came from:

- missing theorems in ZArith, especially with combinations of ordering with $+, \times$
- duplication of symbols ($\leq, \geq$ and $<, >$) which gives rise to several conversions
- complex internal representation of integers (need to manually unfold Ztwo, Zone, ...)
- proof of recursive functions seems to be really tricky

Conclusions (cont'd)

• no chance to overcome some assumption with a system like Coq

• simplifying hypotheses can help to prove more general theorems (ex. $B^h \to H$, ...)

• a nice interface like Pcoq is very useful to display mathematical notations and helps with "mouse-driven" proofs

What remains to do

• so far we've proven only the "mathematical level"

• it remains to prove the real implementation, i.e. to take into account the memory management (how integers are physically represented, check for carries, check that no cell is overwritten before used)

• remains to prove the three other functions `mpn_sqrtrem1`, `mpn_sqrtrem2`, `mpn_sqrtrem_main`