SRI International

THE MECHANICAL VERIFICATION OF A FORTRAN SQUARE ROOT PROGRAM

Robert S. Boyer
J Strother Moore

Computer Science Laboratory
SRI International
Menlo Park, CA 94025

January, 1981

SRI International®

333 Ravenswood Ave. • Menlo Park, California 94025
(415) 326-6200 • Cable: SRI INTL MPK • TWX: 910-373-1246

I THE ALGORITHM*

We shall present a detailed proof of the correctness of a FORTRAN program for computing integer square roots. A similar proof has been mechanically checked by the current version of the theorem-prover described in [2]. The computer system that generates the conjectures to be proved, given the code and assertions, is described in [3].

Here is the FORTRAN code:

```
      INTEGER FUNCTION ISQRT(I)
      INTEGER I
C     CALCULATE THE SQUARE ROOT OF I.
      IF ((I .LT. 0)) STOP
      IF ((I .GT. 1)) GOTO 100
      ISQRT = I
      RETURN
C     ISQRT TAKES ON INCREASINGLY SMALLER VALUES AND CONVERGES TO THE
C     SQUARE ROOT OF I. THE FIRST APPROXIMATION IS ONE HALF I, WHICH IS
C     NOT LESS THAN THE SQUARE ROOT OF I WHEN 1 IS LESS THAN I.
100   ISQRT = (I / 2)
200   CONTINUE
C     ASSERTION LP
      IF (((I / ISQRT) .GE. ISQRT)) RETURN
      ISQRT = ((ISQRT + (I / ISQRT)) / 2)
      GOTO 200
      END
```

To prove ISQRT correct, we start with a specification, which consists of an input and an output assertion. The input assertion states the conditions under which ISQRT may be called; the output assertion states the results that may be expected from an invocation of ISQRT satisfying the input assertion.

---

* The research reported here was supported in part by NASA Contract NAS1-15528, NSF Grant MCS-7904081, and ONR Contract N00014-75-C-0816.

*Input
The value of the input variable I is nonnegative
and is less than the "least inexpressible
positive integer."

What does "the least inexpressible positive integer" mean? The standard definitions of FORTRAN, [4] and [1], fail to specify the precision of arithmetic operations. Thus, strictly speaking, it is extremely difficult to prove anything interesting about a FORTRAN program that uses arithmetic operations. However, in [3] we present some very conservative assumptions about arithmetic operations, and it is upon those assumptions that we shall construct our proof. We assume that for each FORTRAN processor, there exist two integers "the least inexpressible positive integer" (greater than 200) and "the greatest inexpressible negative integer" (less than -200) such that when the processor performs an arithmetic operation (+,*,-,/) upon two INTEGER variables whose "mathematical" answer is strictly between the two inexpressible integers, then the FORTRAN processor returns the correct result.* For example, if the least inexpressible positive integer for a particular FORTRAN processor were 201, then that processor would correctly add 50 and 60, but might not correctly add 100 and 101.

Here is the output assertion:

*Output
If i is the value of the variable I on input and
j is the value returned by the subprogram ISQRT, then

$$j^{**}2 \leq i < (j+1)^{**}2 \text{ and } 0 \leq j.$$

The method we shall use to prove the correctness of ISQRT is the well-known inductive assertion method. In choosing assertions for nodes of the flow graph of ISQRT, we seek propositions about (a) the "initial" values of the input variables, and (b) the "current" values of the input

---

* In the case of division, we make this assumption only if the divisor is not 0.

and local variables such that whenever execution of ISQRT reaches an annotated node, the assertion at that node will be true. Thus, we annotate the first executable statement of ISQRT with the input assertion, we annotate each of the RETURN statements with the output assertion, and we annotate the STOP statement with the false proposition, indicating that we hope never to reach that statement. Finally, we annotate the CONTINUE statement at 200 with the assertion:

```
*lp
if isqrt is the current value of ISQRT and i is the
initial value of I, then

  *a
  0 < isqrt,

  *b
  i ≥ 2*isqrt, and

  *c
  i < (isqrt+1)**2.
```

The choice of *lp is motivated by our task:

> Consider each path through the flow graph of ISQRT whose first and last nodes are annotated, but which contains no intermediate annotated node. For each such path, show that if a FORTRAN processor executes along the path, and if the assertion of the first node is true at the beginning of execution, then when the last node is executed, the assertion there will also be true.

II FROM 200 TO 200

The most interesting path in ISQRT goes from statement 200 to statement 200. Let us walk through the path, assuming that at the beginning *1p holds.

The Logical IF statement involves a division (I / ISQRT). For the division to be legal in FORTRAN, the divisor must be nonzero and the mathematical value of the division must be expressible. Clause *a of *1p assures us that the divisor is positive. To prove that the mathematical value of the division is expressible, note that i, the current value of I, is the same as the input value of I, because I is nowhere altered, and the input value of I is nonnegative and less than the least inexpressible positive integer.

We are supposing that the test of the Logical IF evaluates to false and we proceed to the assignment statement. The proof of the legality of the division (I / ISQRT) is the same as before.

The legality of the addition requires more thought. From clause *b of *1p, we know that 2*isqrt $\leq$ i < the least inexpressible positive integer. Thus, to show that (isqrt + (i / isqrt)) is less than the least inexpressible positive integer, it is sufficient to check that (i / isqrt) < isqrt. But this condition is precisely what we are assured by the falsity of the test of the Logical IF. (That the sum is greater than the greatest inexpressible negative integer follows from the fact that both addends are nonnegative.) The legality of the division by 2 follows from the expressiblity and nonnegativity of the sum.

We now reach 200 again and must prove the the assertion there is true. Let isqrt' = ((isqrt + (i / isqrt)) / 2) and let *1p' be the result of substituting isqrt' for isqrt in *1p. We must prove *1p'. First, let us consider

*a'
0 < isqrt'.

We have that isqrt is positive and i > isqrt. If isqrt is 1, then isqrt' $\geq$ ((1 + (i / 1)) / 2) $\geq$ 1. If isqrt is 2 or greater, then isqrt' $\geq$ (2 / 2) = 1.

Now let us consider

*b'
i $\geq$ 2*isqrt'.

To establish *b', we prove that isqrt > isqrt'.* *b' is then a consequence of *b: i $\geq$ 2*isqrt. First observe that for all nonnegative integers j,

(2 * (j / 2)) $\leq$ j.

(Keep in mind that we are dealing with integer division throughout this note; such apparently trivial observations are worth making explicit.) Using this observation, but replacing j with (isqrt + i/isqrt), and the fact that the test of the Logical IF was false gives us

2*((isqrt + i/isqrt)/2) $\leq$ (isqrt + i/isqrt) < 2*isqrt

from which we conclude that

(isqrt + i/isqrt)/2 < isqrt.

Finally, we turn to the crucial step:

*c'
i < (isqrt'+1)**2.

Let k be the quotient of i by isqrt, and let r be the remainder. Thus, we wish to prove:

---

* This observation is sufficient to guarantee that ISQRT always terminates.

```
k*isqrt + r < ((isqrt + k)/2 + 1)**2.
```

Since r < isqrt, it is sufficient to prove

```
*main
k*isqrt + isqrt ≤ ((isqrt + k)/2 +1)**2.
```

We prove this for all nonnegative k and isqrt by simultaneous induction on k and isqrt. If isqrt is 0, the proof is immediate. So assume isqrt is nonzero. If k is 0, we must show:

```
isqrt ≤ (isqrt/2 + 1)**2,
```

which is equivalent to

```
isqrt ≤ (isqrt/2)**2 + 2*(isqrt/2) + 1.
```

But this is a consequence of the theorem that for all nonnegative j, j-1 ≤ 2*(j/2).

Returning to the inductive step of the proof of *main, we let k and isqrt be nonnegative, we assume

```
k*isqrt + isqrt ≤ ((isqrt + k)/2 +1)**2,
```

i.e.,

```
*hyp
k*isqrt + isqrt ≤ ((isqrt + k)/2)**2 + 2*((isqrt + k)/2) + 1,
```

and we wish to show

```
*concl
(k+1)*(isqrt+1) + (isqrt+1) ≤ (((isqrt+1) + (k+1))/2 +1)**2.
```

We reduce *concl to

```
k*isqrt + k + 2*isqrt + 2  ≤  (1 + (isqrt+k)/2 + 1)**2
```

and then to

```
k*isqrt + k + 2*isqrt+2 ≤ ((isqrt+k)/2)**2 + 4*((isqrt+k)/2)+4.
```

Cancelling with *hyp, we then reduce our goal to

```
k + isqrt ≤ 2*((isqrt+k)/2) + 1,
```

which is a consequence of the theorem that for all nonnegative j, j-1 ≤ 2*(j/2).

III FROM THE BEGINNING TO 200

Consider the path from the beginning to statement 200. When we arrive at statement 200, we know that i, the input value of the variable I, is greater than 1. Let isqrt be (i/2). We need to check

```
*a
0 < isqrt,

*b
i ≥ 2*isqrt, and

*c
i < (isqrt+1)**2.
```

The first two are easy. The last, when the right-hand side is expanded, is

```
i < (i/2)**2 + 2*(i/2) +1.
```

Since i is greater than 1, (i/2) is greater than or equal to 1, and thus the fact that $j-1 \leq 2*(j/2)$ may again be used.

IV FROM 200 TO THE END

Consider the path from statement 200 to the exit of ISQRT. Letting i and isqrt denote the values of the variables I and ISQRT, we may assume

*1p
*a
0 < isqrt,

*b
i $\geq$ 2*isqrt, and

*c
i < (isqrt+1)**2.

Furthermore, from the fact that the test in the Logical IF was true, we have i/isqrt $\geq$ isqrt. We wish to check the output assertion: isqrt**2 $\leq$ i < (isqrt+1)**2 and 0 $\leq$ isqrt. The second conjunct is a consequence of *a. The second inequality of the first conjunct is *c. The first inequality is a consequence of the test and the theorem that for all nonnegative integers, with b nonzero,

a/b $\geq$ c -> a $\geq$ b*c.

To prove that theorem, let k be the quotient of a divided by b and let r be the remainder. The theorem then becomes

k $\geq$ c -> k*b + r $\geq$ b*c,

which follows from the nonnegativity of r and the observation that

k $\geq$ c -> k*b $\geq$ b*c.

V THE OTHER PATHS

We mention the remaining trivial paths only for completion. The STOP statement is never reached because the input condition is that the value of I is nonnegative.

The first RETURN is reached only if the value of I is 0 or 1, in which case, the value of I is its own square root.

VI THE HUMAN BURDEN OF MECHANICAL PROVING

We have proved that the FORTRAN program ISQRT computes the correct answer, always terminates, and never causes a runtime error when called on the specified input.

Because proofs are tedious and error prone when constructed by hand, we used a mechanical theorem-prover to prove the correctness of ISQRT. The theorem-prover was entirely responsible for the accuracy of the proofs. However, the human user occasionally guided the theorem-prover. This guidance took three forms (a) defining the necessary concepts, (b) suggesting important theorems to prove first, and (c) supplying "hints" on how to prove the theorem the machine otherwise has difficulty with. Here are the definitions, theorems, and hints used to make our system prove the correctness of ISQRT.

1. Definition.
   (SQ I)
   =
   (TIMES I I)
   Comment: In the specification of ISQRT and in the invariant, we actually write (SQ X) instead of X**2.

2. Theorem. SQ.REWRITE: (rewrite)
   (EQUAL (SQ I) (TIMES I I))
   Comment: We prove this lemma so we can turn on and off the definition of SQ.

3. Hint: Turn off the definition of SQ.

   Comment: This prevents the definition of SQ from being used, but does not necessarily prevent the lemma SQ.REWRITE from being used.

4. Theorem. PLUS.1: (rewrite)
(EQUAL (PLUS 1 X) (ADD1 X))
Comment: In the latest version of the theorem prover, we have abandoned the heuristic of opening up recursive function definitions merely because a "controller" is an explicit value. While (PLUS 1 X) should probably always be expanded, we have not yet found a good place to draw the line to prohibit the expansion of (PLUS 100000 X).

5. Theorem. DIFFERENCE.2: (rewrite)
(EQUAL (DIFFERENCE (ADD1 (ADD1 X)) 2)
(FIX X))

6. Theorem. QUOTIENT.BY.2: (rewrite)
(NOT (LESSP (PLUS (QUOTIENT A 2) (QUOTIENT A 2))
(SUB1 A)))

7. Theorem. MAIN.TRICK: (rewrite)
(NOT (LESSP (SQ (ADD1 (QUOTIENT (PLUS J K) 2)))
(PLUS (TIMES J K) J)))
Hint: Induct on J and K simultaneously.

8. Theorem. LESSP.REMAINDER2: (rewrite generalize)
(EQUAL (LESSP (REMAINDER X Y) Y)
(NOT (ZEROP Y)))

9. Theorem. REMAINDER.QUOTIENT.ELIM: (elim)
(IMPLIES (AND (NOT (ZEROP Y)) (NUMBERP X))
(EQUAL (PLUS (REMAINDER X Y)
(TIMES Y (QUOTIENT X Y)))
X))

10. Theorem. SQ.ADD1.NON.ZERO: (rewrite)
(NOT (EQUAL (SQ (ADD1 X)) 0))

11. Hint: Turn off the rewrite rule SQ.REWRITE.

12. Theorem. MAIN: (rewrite)
(IMPLIES (NOT (ZEROP J))
(LESSP I
(SQ (ADD1 (QUOTIENT (PLUS J (QUOTIENT I J))
2)))))

13. Hint: Turn the rewrite rule SQ.REWRITE back on again.

14. Theorem. LESSP.TIMES.CANCELLATION.RESTATED.FOR.LINEAR: (rewrite)
(IMPLIES (NOT (LESSP I J))
(NOT (LESSP (TIMES A I) (TIMES A J))))

15. Theorem. MULTIPLY.THRU.BY.DIVISOR: (rewrite)
(IMPLIES (LESSP A (TIMES B C))
(EQUAL (LESSP (QUOTIENT A B) C) T))

16. Theorem. TIMES.GREATERP.ZERO: (rewrite)
(IMPLIES (AND (NOT (ZEROP X)) (NOT (ZEROP Y)))
(LESSP 0 (TIMES X Y)))

17. Theorem. QUOTIENT.SHRINKS: (rewrite)
(NOT (LESSP I (QUOTIENT I J)))

18. Theorem. QUOTIENT.SHRINKS.FAST: (rewrite)
(NOT (LESSP I (TIMES 2 (QUOTIENT I 2))))

19. Theorem. QUOTIENT.BY.1: (rewrite)
(EQUAL (QUOTIENT I 1) (FIX I))

Comment: All the foregoing events occur before any of the verification conditions of ISQRT are attacked. In the proofs of the verification conditions, just before we attack the proof of *1p', we

20. Hint: Turn off the rewrite rule SQ.REWRITE, again!

REFERENCES

1. American National Standards Institute, Inc., American National Standard Programming Language FORTRAN, ANSI X3.9-1978, 1430 Broadway, New York, New York 10018, April 3, 1978.

2. R. S. Boyer and J S. Moore, A Computational Logic, (Academic Press, New York, New York, 1979).

3. Robert S. Boyer and J Strother Moore, "A Verification Condition Generator for FORTRAN," in The Correctness Problem in Computer Science, (eds. Robert S. Boyer and J Strother Moore), to appear, Academic Press, 1981. [SRI International, Computer Science Laboratory Technical Report CSL-103.]

4. United States of America Standards Institute, USA Standard FORTRAN, USAS X3.9-1966, 10 East 40th Street, New York, New York 10016, March 7, 1966.