```
#|

 Copyright (C) 1994 by Robert S. Boyer and J Strother Moore.  All Rights
 Reserved.

 This script is hereby placed in the public domain, and therefore unlimited
 editing and redistribution is permitted.

 NO WARRANTY

 Robert S. Boyer and J Strother Moore PROVIDE ABSOLUTELY NO WARRANTY.  THE
 EVENT SCRIPT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS
 OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF
 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO
 THE QUALITY AND PERFORMANCE OF THE SCRIPT IS WITH YOU.  SHOULD THE SCRIPT
 PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR
 CORRECTION.

 IN NO EVENT WILL Robert S. Boyer or J Strother Moore BE LIABLE TO YOU FOR ANY
 DAMAGES, ANY LOST PROFITS, LOST MONIES, OR OTHER SPECIAL, INCIDENTAL OR
 CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THIS SCRIPT
 (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE
 OR LOSSES SUSTAINED BY THIRD PARTIES), EVEN IF YOU HAVE ADVISED US OF THE
 POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

|#

; This is the list of verification conditions for a FORTRAN square
; root program.  For the details of the algorithm, see the comment at
; the end of the file.  Boyer and Moore.

; This list of events has been further edited, for processing by
; DO-FILE, by (1) inserting the following NOTE-LIB, (2) commenting out
; each FORTRAN-COMMENT and following the comment with the
; corresponding macroexpansion, and (3) by commenting out each
; (COMMENT ...).
```

EVENT: Start with the library `"fortran"` using the compiled version.

```
; (FORTRAN-COMMENT FORTRAN)
```

AXIOM: fortran
t

EVENT: Introduce the function symbol *i$0* of 0 arguments.

EVENT: Introduce the function symbol *isqrt$1* of 0 arguments.

EVENT: Introduce the function symbol *isqrt$2* of 0 arguments.

DEFINITION: $\mathrm{sq}\,(i) = (i * i)$

AXIOM: input-conditions
'*1*true

DEFINITION:
GLOBAL-HYPS
$= \quad ((\mathrm{I\$0} \in \mathbf{N})$
$\quad \wedge \quad ((\mathrm{I\$0} < \text{LEAST-INEXPRESSIBLE-POSITIVE-INTEGER})$
$\quad\quad \wedge \quad \mathrm{znumberp}\,(\mathrm{I\$0})))$

THEOREM: plus-1
$(1 + x) = (1 + x)$

THEOREM: difference-2
$((1 + (1 + x)) - 2) = \mathrm{fix}\,(x)$

THEOREM: quotient-by-2
$((a \div 2) + (a \div 2)) \not< (a - 1)$

THEOREM: main-trick
$\mathrm{sq}\,(1 + ((j + k) \div 2)) \not< ((j * k) + j)$

THEOREM: lessp-remainder2
$((x \textbf{ mod } y) < y) = (y \not\simeq 0)$

THEOREM: remainder-quotient-elim
$((y \not\simeq 0) \wedge (x \in \mathbf{N})) \rightarrow (((x \textbf{ mod } y) + (y * (x \div y))) = x)$

THEOREM: sq-add1-non-zero
$\mathrm{sq}\,(1 + x) \neq 0$

EVENT: Disable sq.

THEOREM: main
$(j \not\simeq 0) \rightarrow (i < \mathrm{sq}\,(1 + ((j + (i \div j)) \div 2)))$

EVENT: Enable sq.

THEOREM: lessp-times-cancellation-restated-for-linear
$(i \not< j) \rightarrow ((a * i) \not< (a * j))$

THEOREM: multiply-thru-by-divisor
$(a < (b * c)) \rightarrow (((a \div b) < c) = \mathbf{t})$

THEOREM: times-greaterp-zero
$((x \not\simeq 0) \land (y \not\simeq 0)) \rightarrow (0 < (x * y))$

THEOREM: quotient-shrinks
$i \not< (i \div j)$

THEOREM: quotient-shrinks-fast
$i \not< (2 * (i \div 2))$

THEOREM: quotient-by-1
$(i \div 1) = \text{fix}(i)$

```
; (FORTRAN-COMMENT INPUT)
```

AXIOM: input
t

```
; (FORTRAN-COMMENT LOGICAL-IF-T)
```

AXIOM: logical-if-t
t

THEOREM: stop
$(\neg\ \text{zlessp}(\text{I\$0},\ \texttt{'0})) \lor (\neg\ \text{GLOBAL-HYPS})$

```
#| (COMMENT INPUT T) |#
```

EVENT: Undo back through the event named 'logical-if-t'.

```
; (FORTRAN-COMMENT LOGICAL-IF-F)
```

AXIOM: logical-if-f
t

```
; (FORTRAN-COMMENT LOGICAL-IF-T)
```

AXIOM: logical-if-t
t

THEOREM: input-cond-of-zquotient
$(\text{zgreaterp}(\textsc{i}\$0, \texttt{'1}) \wedge \textsc{global-hyps})$
$\rightarrow \quad ((\neg\ \text{zeqp}(\texttt{'2}, \texttt{'0})) \wedge \text{expressible-znumberp}(\text{zquotient}(\textsc{i}\$0, \texttt{'2})))$

```
#| (COMMENT INPUT F T) |#
```

AXIOM: assignment
$\textsc{isqrt}\$1 = \text{zquotient}(\textsc{i}\$0, \texttt{'2})$

THEOREM: lp
$(\text{zgreaterp}(\textsc{i}\$0, \texttt{'1}) \wedge \textsc{global-hyps})$
$\rightarrow \quad (((\texttt{'0} < \textsc{isqrt}\$1)$
$\quad \wedge \quad ((\textsc{i}\$0 \not< (\texttt{'2} * \textsc{isqrt}\$1))$
$\quad\quad \wedge \quad ((\textsc{isqrt}\$1 \in \mathbf{N}) \wedge (\textsc{i}\$0 < \text{sq}(1 + \textsc{isqrt}\$1)))))$
$\quad \wedge \quad \text{lex}(\text{cons}(\textsc{isqrt}\$1, \texttt{'nil}), \text{cons}(\textsc{i}\$0, \texttt{'nil})))$

```
#| (COMMENT INPUT F T) |#
```

EVENT: Undo back through the event named ‘logical-if-t’.

```
; (FORTRAN-COMMENT LOGICAL-IF-F1)
```

AXIOM: logical-if-f1
t

AXIOM: assignment
$\textsc{isqrt}\$1 = \textsc{i}\$0$

THEOREM: output
$((\neg\ \text{zgreaterp}(\textsc{i}\$0, \texttt{'1})) \wedge \textsc{global-hyps})$
$\rightarrow \quad (\text{znumberp}(\textsc{isqrt}\$1)$
$\quad \wedge \quad (\text{zgreatereqp}(\textsc{isqrt}\$1, \texttt{'0})$
$\quad\quad \wedge \quad ((\textsc{i}\$0 \not< \text{sq}(\textsc{isqrt}\$1))$
$\quad\quad\quad \wedge \quad (\textsc{i}\$0 < \text{sq}(\texttt{'1} + \textsc{isqrt}\$1)))))$

```
#| (COMMENT INPUT F F) |#
```

EVENT: Undo back through the event named ‘input’.

AXIOM: paths-from-lp
`'*1*true`

Definition:
PATH-HYPS
$= \quad$ (GLOBAL-HYPS
$\wedge \quad$ ((`'0` $<$ ISQRT\$1)
$\wedge \quad$ ((I\$0 $\not<$ (`'2` $*$ ISQRT\$1))
$\wedge \quad$ ((ISQRT\$1 $\in \mathbf{N}$) $\wedge$ (I\$0 $<$ sq (1 + ISQRT\$1))))))

Theorem: definedness
PATH-HYPS $\rightarrow$ znumberp (ISQRT\$1)

```
#| (COMMENT LP) |#
```

Theorem: input-cond-of-zquotient
PATH-HYPS
$\rightarrow \quad$ (($\neg$ zeqp (ISQRT\$1, `'0`))
$\wedge \quad$ expressible-znumberp (zquotient (I\$0, ISQRT\$1)))

```
#| (COMMENT LP) |#

; (FORTRAN-COMMENT LOGICAL-IF-T)
```

Axiom: logical-if-t
t

Theorem: output1
(zgreatereqp (zquotient (I\$0, ISQRT\$1), ISQRT\$1) $\wedge$ PATH-HYPS)
$\rightarrow \quad$ (znumberp (ISQRT\$1)
$\wedge \quad$ (zgreatereqp (ISQRT\$1, `'0`)
$\wedge \quad$ ((I\$0 $\not<$ sq (ISQRT\$1))
$\wedge \quad$ (I\$0 $<$ sq (`'1` + ISQRT\$1)))))

```
#| (COMMENT LP T) |#
```

Event: Undo back through the event named 'logical-if-t'.

```
; (FORTRAN-COMMENT LOGICAL-IF-F2)
```

Axiom: logical-if-f2
t

Theorem: input-cond-of-zquotient1
(($\neg$ zgreatereqp (zquotient (I\$0, ISQRT\$1), ISQRT\$1)) $\wedge$ PATH-HYPS)
$\rightarrow \quad$ (($\neg$ zeqp (ISQRT\$1, `'0`))
$\wedge \quad$ expressible-znumberp (zquotient (I\$0, ISQRT\$1)))

```
#| (COMMENT LP F) |#
```

THEOREM: input-cond-of-zplus
$((\neg \text{zgreatereqp}(\text{zquotient}(\textsc{i}\$0, \textsc{isqrt}\$1), \textsc{isqrt}\$1)) \land \textsc{path-hyps})$
$\rightarrow \quad \text{expressible-znumberp}(\text{zplus}(\textsc{isqrt}\$1, \text{zquotient}(\textsc{i}\$0, \textsc{isqrt}\$1)))$

```
#| (COMMENT LP F) |#
```

THEOREM: input-cond-of-zquotient2
$((\neg \text{zgreatereqp}(\text{zquotient}(\textsc{i}\$0, \textsc{isqrt}\$1), \textsc{isqrt}\$1)) \land \textsc{path-hyps})$
$\rightarrow \quad ((\neg \text{zeqp}(\texttt{'2}, \texttt{'0}))$
$\quad \land \quad \text{expressible-znumberp}(\text{zquotient}(\text{zplus}(\textsc{isqrt}\$1,$
$\qquad \text{zquotient}(\textsc{i}\$0, \textsc{isqrt}\$1)),$
$\qquad \texttt{'2})))$

```
#| (COMMENT LP F) |#
```

AXIOM: assignment1
$\textsc{isqrt}\$2 = \text{zquotient}(\text{zplus}(\textsc{isqrt}\$1, \text{zquotient}(\textsc{i}\$0, \textsc{isqrt}\$1)), \texttt{'2})$

EVENT: Disable sq.

THEOREM: lp1
$((\neg \text{zgreatereqp}(\text{zquotient}(\textsc{i}\$0, \textsc{isqrt}\$1), \textsc{isqrt}\$1)) \land \textsc{path-hyps})$
$\rightarrow \quad (((\texttt{'0} < \textsc{isqrt}\$2)$
$\quad \land \quad ((\textsc{i}\$0 \not< (\texttt{'2} * \textsc{isqrt}\$2))$
$\qquad \land \quad ((\textsc{isqrt}\$2 \in \mathbf{N}) \land (\textsc{i}\$0 < \text{sq}(1 + \textsc{isqrt}\$2)))))$
$\quad \land \quad \text{lex}(\text{cons}(\textsc{isqrt}\$2, \texttt{'nil}), \text{cons}(\textsc{isqrt}\$1, \texttt{'nil})))$

```
#| (COMMENT LP F) |#
```

EVENT: Undo back through the event named 'paths-from-lp'.

EVENT: Undo back through the event named 'fortran'.

```
#|
```

```
The correctness of the program depends upon the following events:
```

```
@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Definition.
        (SQ I)
           =
        (TIMES I I)
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        (FORTRAN-COMMENT ISQRT-STUFF)
@END(VERBATIM)
@END(GROUP)
```

Specification for routine ISQRT

The input assertion:

```
        (AND (NUMBERP (I STATE))
             (LESSP (I STATE)
                    (LEAST-INEXPRESSIBLE-POSITIVE-INTEGER)))
```

The output assertion:

```
        (AND (ZGREATEREQP ANS 0)
             (NOT (LESSP (I STATE) (SQ ANS)))
             (LESSP (I STATE) (SQ (PLUS 1 ANS))))
```

```
      END
      INTEGER FUNCTION ISQRT(I)
      INTEGER I
C     CALCULATE THE SQUARE ROOT OF I USING THE NEWTON METHOD.
      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 SQ
C     UARE ROOT OF I. THE FIRST APPROXIMATION IS ONE HALF I, WHICH IS NO
C     T 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)
C     XXX SQ-REWRITE-OFF-AGAIN
      GOTO 200
      END
```

```
The XXX at ISQRT-STUFF.
@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  PLUS-1 (rewrite):
        (EQUAL (PLUS 1 X) (ADD1 X))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  DIFFERENCE-2 (rewrite):
        (EQUAL (DIFFERENCE (ADD1 (ADD1 X)) 2)
               (FIX X))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  QUOTIENT-BY-2 (rewrite):
        (NOT (LESSP (PLUS (QUOTIENT A 2) (QUOTIENT A 2))
                    (SUB1 A)))
@END(VERBATIM)
```

```
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  MAIN-TRICK (rewrite):
        (NOT (LESSP (SQ (ADD1 (QUOTIENT (PLUS J K) 2)))
                    (PLUS (TIMES J K) J)))
        Hint:  Induct as for (LESSP J K).
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  LESSP-REMAINDER2 (rewrite and generalize):
        (EQUAL (LESSP (REMAINDER X Y) Y)
               (NOT (ZEROP Y)))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  REMAINDER-QUOTIENT-ELIM (elimination):
        (IMPLIES (AND (NOT (ZEROP Y)) (NUMBERP X))
                 (EQUAL (PLUS (REMAINDER X Y)
                              (TIMES Y (QUOTIENT X Y)))
                        X))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  SQ-ADD1-NON-ZERO (rewrite):
        (NOT (EQUAL (SQ (ADD1 X)) 0))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Enable SQ.
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
```

```
        Theorem.  MAIN (rewrite):
        (IMPLIES (NOT (ZEROP J))
                 (LESSP I
                        (SQ (ADD1 (QUOTIENT (PLUS J (QUOTIENT I J))
                                            2)))))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Disable SQ.
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  LESSP-TIMES-CANCELLATION-RESTATED-FOR-LINEAR (rewrite):
        (IMPLIES (NOT (LESSP I J))
                 (NOT (LESSP (TIMES A I) (TIMES A J))))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  MULTIPLY-THRU-BY-DIVISOR (rewrite):
        (IMPLIES (LESSP A (TIMES B C))
                 (EQUAL (LESSP (QUOTIENT A B) C) T))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  TIMES-GREATERP-ZERO (rewrite):
        (IMPLIES (AND (NOT (ZEROP X)) (NOT (ZEROP Y)))
                 (LESSP 0 (TIMES X Y)))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  QUOTIENT-SHRINKS (rewrite):
        (NOT (LESSP I (QUOTIENT I J)))
@END(VERBATIM)
@END(GROUP)
```

```
@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  QUOTIENT-SHRINKS-FAST (rewrite):
        (NOT (LESSP I (TIMES 2 (QUOTIENT I 2))))
@END(VERBATIM)
@END(GROUP)

@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Theorem.  QUOTIENT-BY-1 (rewrite):
        (EQUAL (QUOTIENT I 1) (FIX I))
@END(VERBATIM)
@END(GROUP)
```

```
Hints for routine ISQRT



The input clock:
        (LIST (I (START)))

The invariant and clock named LP.

        (AND (LESSP 0 (ISQRT STATE))
             (NOT (LESSP (I STATE)
                         (TIMES 2 (ISQRT STATE))))
             (NUMBERP (ISQRT STATE))
             (LESSP (I STATE)
                    (SQ (ADD1 (ISQRT STATE)))))

        (LIST (ISQRT STATE))

The XXX named SQ-REWRITE-OFF-AGAIN:
@BEGIN(GROUP)
@BEGIN(VERBATIM)
        Enable SQ.
@END(VERBATIM)
@END(GROUP)

|#
```

Index