```
#|

 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.

|#
```

EVENT: Start with the initial **nqthm** theory.

DEFINITION: $\text{logicalp}(x) = ((x = \text{TRUE}) \vee (x = \text{FALSE}))$

DEFINITION:
$\text{expt}(i, j)$
$=$ **if** $j \simeq 0$ **then** 1
else $i * \text{expt}(i, j - 1)$ **endif**

DEFINITION: $\text{znumberp}(x) = (\text{negativep}(x) \vee (x \in \mathbf{N}))$

DEFINITION: $\text{ZZERO} = \text{ZERO}$

DEFINITION:
$\text{zplus}(x, y)$
$=$ **if** $\text{negativep}(x)$

then if negativep (y) **then** $-$ (negative-guts (x) + negative-guts (y))
elseif $y <$ negative-guts (x) **then** $-$ (negative-guts (x) $-$ y)
else $y -$ negative-guts (x) **endif**
elseif negativep (y)
then if $x <$ negative-guts (y) **then** $-$ (negative-guts (y) $-$ x)
else $x -$ negative-guts (y) **endif**
else $x + y$ **endif**

DEFINITION:
zdifference $(x,\ y)$
$=$ **if** negativep (x)
then if negativep (y)
then if negative-guts $(y) <$ negative-guts (x)
then $-$ (negative-guts (x) $-$ negative-guts (y))
else negative-guts (y) $-$ negative-guts (x) **endif**
else $-$ (negative-guts (x) + y) **endif**
elseif negativep (y) **then** $x +$ negative-guts (y)
elseif $x < y$ **then** $-$ $(y - x)$
else $x - y$ **endif**

DEFINITION:
ztimes $(x,\ y)$
$=$ **if** negativep (x)
then if negativep (y) **then** negative-guts $(x) *$ negative-guts (y)
else $-$ (negative-guts $(x) * y$) **endif**
elseif negativep (y) **then** $-$ $(x *$ negative-guts $(y))$
else $x * y$ **endif**

DEFINITION:
zquotient $(x,\ y)$
$=$ **if** negativep (x)
then if negativep (y) **then** negative-guts $(x) \div$ negative-guts (y)
else $-$ (negative-guts $(x) \div y$) **endif**
elseif negativep (y) **then** $-$ $(x \div$ negative-guts $(y))$
else $x \div y$ **endif**

DEFINITION:
zexptz $(i,\ j)$
$=$ **if** $j \simeq$ `0` **then** `1`
else ztimes $(i,$ zexptz $(i,\ j - 1))$ **endif**

DEFINITION:
znormalize (x)
$=$ **if** negativep (x)
then if negative-guts $(x) =$ `0` **then** `0`

else x **endif**
else fix (x) **endif**

DEFINITION: zeqp (x, y) = (znormalize (x) = znormalize (y))

DEFINITION: zneqp (x, y) = ($\neg$ zeqp (x, y))

DEFINITION:
zlessp (x, y)
= **if** negativep (x)
then if negativep (y) **then** negative-guts (y) < negative-guts (x)
else $\neg$ ((negative-guts (x) = 0) $\wedge$ ($y \simeq$ 0)) **endif**
elseif negativep (y) **then f**
else $x < y$ **endif**

DEFINITION: zlesseqp (x, y) = ($\neg$ zlessp (y, x))

DEFINITION: zgreaterp (x, y) = zlessp (y, x)

DEFINITION: zgreatereqp (x, y) = ($\neg$ zlessp (x, y))

CONSERVATIVE AXIOM: integer-size
(LEAST-INEXPRESSIBLE-POSITIVE-INTEGER $\in$ **N**)
$\wedge$ negativep (GREATEST-INEXPRESSIBLE-NEGATIVE-INTEGER)
$\wedge$ (200 < negative-guts (GREATEST-INEXPRESSIBLE-NEGATIVE-INTEGER))
$\wedge$ (200 < LEAST-INEXPRESSIBLE-POSITIVE-INTEGER)

Simultaneously, we introduce the new function symbols *greatest-inexpressible-negative-integer* and *least-inexpressible-positive-integer*.

DEFINITION:
expressible-znumberp (x)
= (zlessp (GREATEST-INEXPRESSIBLE-NEGATIVE-INTEGER, x)
$\wedge$ zlessp (x, LEAST-INEXPRESSIBLE-POSITIVE-INTEGER))

DEFINITION:
iabs (i)
= **if** negativep (i) **then** negative-guts (i)
else fix (i) **endif**

DEFINITION: mod (x, y) = zdifference $(x,$ ztimes $(y,$ zquotient $(x, y)))$

DEFINITION:
max0 (i, j)
= **if** zlessp (i, j) **then** j
else i **endif**

DEFINITION:
min0 (i, j)
= **if** zlessp (i, j) **then** i
else j **endif**

DEFINITION:
isign (i, j)
= **if** negativep (j) **then** ztimes (`-1`, iabs (i))
else iabs (i) **endif**

DEFINITION: idim (i, j) = zdifference (i, min0 (i, j))

```
;;; In the old FORTRAN xxx we used UNDEF for all occurrences of the
;;; substring FORTRAN-UNDEF below.  However, in the quantifier version
;;; of the logic, UNDEF is a function name set up in the bootstrap.
;;; To avoid conflict we have changed the name here.  However, it is
;;; possible that vcs mention UNDEF -- not just the function DEFINEDP
;;; which is still around here.  If so, the vcg will have be to be
;;; changed should we decide to support it.
```

EVENT: Add the shell *fortran-undef*, with recognizer function symbol *fortran-undefined* and 1 accessor: *fortran-undef-guts*, with type restriction (none-of) and default value zero.

DEFINITION: definedp (x) = ($\neg$ fortran-undefined (x))

EVENT: Introduce the function symbol *elt1* of 2 arguments.

EVENT: Introduce the function symbol *elt2* of 3 arguments.

EVENT: Introduce the function symbol *elt3* of 4 arguments.

DEFINITION:
lex (*l1*, *l2*)
= **if** (*l1* $\simeq$ **nil**) $\vee$ (*l2* $\simeq$ **nil**) **then f**
else (car (*l1*) $<$ car (*l2*))
$\vee$ ((car (*l1*) = car (*l2*)) $\wedge$ lex (cdr (*l1*), cdr (*l2*))) **endif**

EVENT: Introduce the function symbol *rnumberp* of one argument.

EVENT: Introduce the function symbol *dnumberp* of one argument.

EVENT: Introduce the function symbol *cnumberp* of one argument.

EVENT: Introduce the function symbol *rzero* of 0 arguments.

EVENT: Introduce the function symbol *dzero* of 0 arguments.

EVENT: Introduce the function symbol *czero* of 0 arguments.

EVENT: Introduce the function symbol *expressible-rnumberp* of one argument.

EVENT: Introduce the function symbol *expressible-dnumberp* of one argument.

EVENT: Introduce the function symbol *expressible-cnumberp* of one argument.

EVENT: Introduce the function symbol *rplus* of 2 arguments.

EVENT: Introduce the function symbol *rtimes* of 2 arguments.

EVENT: Introduce the function symbol *rdifference* of 2 arguments.

EVENT: Introduce the function symbol *rquotient* of 2 arguments.

EVENT: Introduce the function symbol *rlessp* of 2 arguments.

EVENT: Introduce the function symbol *rlesseqp* of 2 arguments.

EVENT: Introduce the function symbol *reqp* of 2 arguments.

EVENT: Introduce the function symbol *rneqp* of 2 arguments.

EVENT: Introduce the function symbol *rgreatereqp* of 2 arguments.

EVENT: Introduce the function symbol *rgreaterp* of 2 arguments.

EVENT: Introduce the function symbol *dplus* of 2 arguments.

EVENT: Introduce the function symbol *dtimes* of 2 arguments.

EVENT: Introduce the function symbol *ddifference* of 2 arguments.

EVENT: Introduce the function symbol *dquotient* of 2 arguments.

EVENT: Introduce the function symbol *dlessp* of 2 arguments.

EVENT: Introduce the function symbol *dlesseqp* of 2 arguments.

EVENT: Introduce the function symbol *deqp* of 2 arguments.

EVENT: Introduce the function symbol *dneqp* of 2 arguments.

EVENT: Introduce the function symbol *dgreatereqp* of 2 arguments.

EVENT: Introduce the function symbol *dgreaterp* of 2 arguments.

EVENT: Introduce the function symbol *cplus* of 2 arguments.

EVENT: Introduce the function symbol *ctimes* of 2 arguments.

EVENT: Introduce the function symbol *cdifference* of 2 arguments.

EVENT: Introduce the function symbol *cquotient* of 2 arguments.

EVENT: Introduce the function symbol *ceqp* of 2 arguments.

EVENT: Introduce the function symbol *cneqp* of 2 arguments.

EVENT: Introduce the function symbol *rexptz* of 2 arguments.

EVENT: Introduce the function symbol *dexptz* of 2 arguments.

EVENT: Introduce the function symbol *cexptz* of 2 arguments.

EVENT: Introduce the function symbol *rexptr* of 2 arguments.

EVENT: Introduce the function symbol *rexptd* of 2 arguments.

EVENT: Introduce the function symbol *dexptr* of 2 arguments.

EVENT: Introduce the function symbol *dexptd* of 2 arguments.

EVENT: Introduce the function symbol *abs* of one argument.

EVENT: Introduce the function symbol *dabs* of one argument.

EVENT: Introduce the function symbol *aint* of one argument.

EVENT: Introduce the function symbol *int* of one argument.

EVENT: Introduce the function symbol *idint* of one argument.

EVENT: Introduce the function symbol *amod* of 2 arguments.

EVENT: Introduce the function symbol *amax0* of 2 arguments.

EVENT: Introduce the function symbol *amax1* of 2 arguments.

EVENT: Introduce the function symbol *max1* of 2 arguments.

EVENT: Introduce the function symbol *dmax1* of 2 arguments.

EVENT: Introduce the function symbol *amin0* of 2 arguments.

EVENT: Introduce the function symbol *amin1* of 2 arguments.

EVENT: Introduce the function symbol *min1* of 2 arguments.

EVENT: Introduce the function symbol *dmin1* of 2 arguments.

EVENT: Introduce the function symbol *float* of one argument.

EVENT: Introduce the function symbol *ifix* of one argument.

EVENT: Introduce the function symbol *sign* of 2 arguments.

EVENT: Introduce the function symbol *dsign* of 2 arguments.

EVENT: Introduce the function symbol *dim* of 2 arguments.

EVENT: Introduce the function symbol *sngl* of one argument.

EVENT: Introduce the function symbol *real* of one argument.

EVENT: Introduce the function symbol *aimag* of one argument.

EVENT: Introduce the function symbol *dble* of one argument.

EVENT: Introduce the function symbol *cmplx* of 2 arguments.

EVENT: Introduce the function symbol *conjg* of one argument.

EVENT: Introduce the function symbol *exp* of one argument.

EVENT: Introduce the function symbol *dexp* of one argument.

EVENT: Introduce the function symbol *cexp* of one argument.

EVENT: Introduce the function symbol *alog* of one argument.

EVENT: Introduce the function symbol *dlog* of one argument.

EVENT: Introduce the function symbol *clog* of one argument.

EVENT: Introduce the function symbol *alog10* of one argument.

EVENT: Introduce the function symbol *dlog10* of one argument.

EVENT: Introduce the function symbol *sin* of one argument.

EVENT: Introduce the function symbol *dsin* of one argument.

EVENT: Introduce the function symbol *csin* of one argument.

EVENT: Introduce the function symbol *cos* of one argument.

EVENT: Introduce the function symbol *dcos* of one argument.

EVENT: Introduce the function symbol *ccos* of one argument.

EVENT: Introduce the function symbol *tanh* of one argument.

EVENT: Introduce the function symbol *sqrt* of one argument.

EVENT: Introduce the function symbol *dsqrt* of one argument.

EVENT: Introduce the function symbol *csqrt* of one argument.

EVENT: Introduce the function symbol *atan* of one argument.

EVENT: Introduce the function symbol *datan* of one argument.

EVENT: Introduce the function symbol *atan2* of 2 arguments.

EVENT: Introduce the function symbol *datan2* of 2 arguments.

EVENT: Introduce the function symbol *dmod* of 2 arguments.

EVENT: Introduce the function symbol *cabs* of one argument.

DEFINITION:

almost-equal1 ($a1$, $a2$, u, v, i, e)
= **if** $(v \simeq \mathtt{0}) \vee (v < u)$ **then t**
else if $v = i$ **then** elt1 ($a2$, v) $= e$
else elt1 ($a2$, v) $=$ elt1 ($a1$, v) **endif**
$\wedge$ almost-equal1 ($a1$, $a2$, u, $v - 1$, i, e) **endif**

THEOREM: plus-0
$(x + \mathtt{0}) = \text{fix}\,(x)$

THEOREM: plus-non-numberp
$(y \notin \mathbf{N}) \rightarrow ((x + y) = \text{fix}\,(x))$

THEOREM: plus-add1
$(x + (1 + y))$
= **if** $y \in \mathbf{N}$ **then** $1 + (x + y)$
else $1 + x$ **endif**

THEOREM: commutativity2-of-plus
$(x + (y + z)) = (y + (x + z))$

THEOREM: commutativity-of-plus
$(x + y) = (y + x)$

THEOREM: associativity-of-plus
$((x + y) + z) = (x + (y + z))$

THEOREM: times-0
$(x * \mathtt{0}) = \mathtt{0}$

THEOREM: times-non-numberp
$(y \notin \mathbf{N}) \rightarrow ((x * y) = \mathtt{0})$

THEOREM: distributivity-of-times-over-plus
$(x * (y + z)) = ((x * y) + (x * z))$

THEOREM: times-add1
$(x * (1 + y))$
= **if** $y \in \mathbf{N}$ **then** $x + (x * y)$
else fix (x) **endif**

THEOREM: commutativity2-of-times
$(x * (y * z)) = (y * (x * z))$

THEOREM: commutativity-of-times
$(x * y) = (y * x)$

THEOREM: associativity-of-times
$((x * y) * z) = (x * (y * z))$

THEOREM: equal-times-0
$((x * y) = \mathtt{0}) = ((x \simeq \mathtt{0}) \vee (y \simeq \mathtt{0}))$

THEOREM: equal-lessp
$((x < y) = z)$
$=$ **if** $x < y$ **then t** $= z$
else f $= z$ **endif**

THEOREM: almost-equal1-in-range
$((\text{elt1}\,(a2,\, j) \neq w)$
$\wedge \quad (w =$ **if** $j = i$ **then** e
else $\text{elt1}\,(a1,\, j)$ **endif**$)$
$\wedge \quad (u \not\simeq \mathtt{0})$
$\wedge \quad (j \not< u)$
$\wedge \quad (v \not< j))$
$\rightarrow \quad (\neg\ \text{almost-equal1}\,(a1,\, a2,\, u,\, v,\, i,\, e))$

THEOREM: almost-equal1-in-range-opened-up
$((\text{elt1}\,(a2,\, j) \neq w)$
$\wedge \quad (w =$ **if** $j = i$ **then** e
else $\text{elt1}\,(a1,\, j)$ **endif**$)$
$\wedge \quad (u \not\simeq \mathtt{0})$
$\wedge \quad (u \leq j)$
$\wedge \quad (j \leq v)$
$\wedge \quad (v \not\simeq \mathtt{0})$
$\wedge \quad (v \not< u)$
$\wedge \quad (v \neq i)$
$\wedge \quad (\text{elt1}\,(a2,\, v) = \text{elt1}\,(a1,\, v)))$
$\rightarrow \quad (\neg\ \text{almost-equal1}\,(a1,\, a2,\, u,\, v - 1,\, i,\, e))$

THEOREM: almost-equal1-contracts
$(\text{almost-equal1}\,(a1,\, a2,\, u,\, v,\, i,\, e) \wedge (u \not\simeq \mathtt{0}) \wedge (x \not< u) \wedge (v \not< y))$
$\rightarrow \quad \text{almost-equal1}\,(a1,\, a2,\, x,\, y,\, i,\, e)$

EVENT: Make the library `"fortran"` and compile it.

Index