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

```
#|
----------------------------------------------------------------------------
Date:      Jan, 1991
Modified:  May 18, 1992.
File:      mc20-0.events
----------------------------------------------------------------------------

                  AN ARITHMETIC SUBLIBRARY
|#

; before we start to write our specification, some preliminary theories
; have to be established.

;              THEOREMS ABOUT SET AND BAG
; subset relation.
```

DEFINITION:
subset $(x,\ y)$
$=$ **if** listp (x)
then if car $(x) \in y$ **then** subset (cdr $(x),\ y)$
else f endif
else t endif

```
; delete the elements of x from the set y.
```

DEFINITION:
delete $(x,\ y)$
$=$ **if** listp (y)
then if $x =$ car (y) **then** cdr (y)
else cons (car (y), delete $(x$, cdr $(y)))$ **endif**
else y **endif**

```
; determines whether x is a subbag of y.
```

DEFINITION:
subbagp $(x,\ y)$
$=$ **if** listp (x)
then if car $(x) \in y$ **then** subbagp (cdr (x), delete (car $(x),\ y))$
else f endif
else t endif

```
; the difference.
```

DEFINITION:
bagdiff $(x,\ y)$
$=$ **if** listp (y)
then if car $(y) \in x$ **then** bagdiff (delete (car $(y),\ x)$, cdr $(y))$
else bagdiff $(x$, cdr $(y))$ **endif**
else x **endif**

```
; the intersection.
```

DEFINITION:
bagint $(x,\ y)$
$=$ **if** listp (x)
then if car $(x) \in y$
then cons (car (x), bagint (cdr (x), delete (car $(x),\ y)))$
else bagint (cdr $(x),\ y)$ **endif**
else nil endif

THEOREM: delete-non-member
$(x \notin y) \rightarrow$ (delete $(x,\ y) = y)$

THEOREM: member-delete
$(x \in \text{delete}(u, v)) \rightarrow (x \in v)$

THEOREM: delete-commutativity
$\text{delete}(x, \text{delete}(y, z)) = \text{delete}(y, \text{delete}(x, z))$

THEOREM: subbagp-delete
$\text{subbagp}(x, \text{delete}(u, y)) \rightarrow \text{subbagp}(x, y)$

THEOREM: subbagp-cdr1
$\text{subbagp}(x, y) \rightarrow \text{subbagp}(\text{cdr}(x), y)$

THEOREM: subbagp-cdr2
$\text{subbagp}(x, \text{cdr}(y)) \rightarrow \text{subbagp}(x, y)$

THEOREM: subbagp-bagint1
$\text{subbagp}(\text{bagint}(x, y), x)$

THEOREM: subbagp-bagint2
$\text{subbagp}(\text{bagint}(x, y), y)$

```
;              THEOREMS ABOUT NATURAL NUMBERS
; lemmas about lessp.
```

THEOREM: lessp-of-1
$(x < \mathtt{1}) = (x \simeq \mathtt{0})$

THEOREM: lessp-sub1
$((x - 1) < x) = (x \not\simeq \mathtt{0})$

```
; lemmas about plus.
```

THEOREM: plus-add1
$(\mathtt{1} + x) = (1 + x)$

THEOREM: plus-add1-1
$(x + (1 + y)) = (1 + (x + y))$

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

THEOREM: plus-commutativity1
$(x + (y + z)) = (y + (x + z))$

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

THEOREM: plus-equal-cancel0
$((x + y) = x) = ((x \in \mathbf{N}) \wedge (y \simeq \mathsf{0}))$

THEOREM: plus-equal-cancel
$((x + y) = (x + z)) = (\mathrm{fix}\,(y) = \mathrm{fix}\,(z))$

THEOREM: plus-lessp-cancel-0
$(x < (x + y)) = (y \not\simeq \mathsf{0})$

THEOREM: plus-lessp-cancel-1
$((x + y) < (x + z)) = (y < z)$

THEOREM: plus-lessp-cancel-add1
$((y + x) < (1 + y)) = (x \simeq \mathsf{0})$

THEOREM: plus-equal-0
$((x + y) = \mathsf{0}) = ((x \simeq \mathsf{0}) \wedge (y \simeq \mathsf{0}))$

```
; lemmas about difference.
```

THEOREM: sub1-of-1
$((x - 1) = \mathsf{0}) = ((x \simeq \mathsf{0}) \vee (x = \mathsf{1}))$

THEOREM: difference-sub1
$(x - \mathsf{1}) = (x - 1)$

THEOREM: difference-sub1-sub1
$((x - (y - 1)) - 1)$
$= \ $ **if** $y \simeq \mathsf{0}$ **then** $x - 1$
elseif $x < y$ **then** $\mathsf{0}$
else $x - y$ **endif**

THEOREM: difference-0
$(x \leq y) \rightarrow ((x - y) = \mathsf{0})$

THEOREM: difference-x-x
$(x - x) = \mathsf{0}$

THEOREM: difference-plus-cancel0
$(((x + y) - x) = \mathrm{fix}\,(y)) \wedge (((y + x) - x) = \mathrm{fix}\,(y))$

THEOREM: difference-plus1
$(x \not< y) \rightarrow (((x - y) + z) = ((x + z) - y))$

THEOREM: difference-plus2
$(y \not< z) \rightarrow ((x + (y - z)) = ((x + y) - z))$

THEOREM: difference-difference1
$((x - y) - z) = (x - (y + z))$

THEOREM: difference-difference2
$(y \not< z) \rightarrow ((x - (y - z)) = ((x + z) - y))$

THEOREM: difference-plus-cancel1
$((x + y) - (x + z)) = (y - z)$

THEOREM: difference-plus-cancel-add1
$((y + x) - (1 + y)) = (x - 1)$

THEOREM: difference-lessp
$((m - n) < m) = ((m \not\simeq \mathtt{0}) \wedge (n \not\simeq \mathtt{0}))$

THEOREM: difference-lessp1
$(x < z) \rightarrow (((x - y) < z) = \mathbf{t})$

THEOREM: difference=0
$(\mathtt{0} = (x - y)) = (y \not< x)$

THEOREM: difference-equal-cancel-0
$(x = (x - y)) = ((x \in \mathbf{N}) \wedge ((x = \mathtt{0}) \vee (y \simeq \mathtt{0})))$

THEOREM: difference-equal-cancel-1
$((x - z) = (y - z))$
$=$ **if** $x < z$ **then** $z \not< y$
elseif $y < z$ **then** $z \not< x$
else $\mathrm{fix}(x) = \mathrm{fix}(y)$ **endif**

THEOREM: difference-lessp-cancel
$((a - c) < (b - c))$
$=$ **if** $c \leq a$ **then** $a < b$
else $c < b$ **endif**

```
; meta lemmas for plus and difference.  Stolen from basic.events.
```

DEFINITION:
plus-fringe (x)
$=$ **if** listp $(x) \wedge (\mathrm{car}(x) =$ `'plus`$)$
then append (plus-fringe (cadr (x)), plus-fringe (caddr (x)))
else cons $(x$, **nil**$)$ **endif**

DEFINITION:
plus-tree (l)
$=$ **if** $l \simeq$ **nil** **then** `''0`
elseif cdr $(l) \simeq$ **nil** **then** list (`'fix`, car (l))
elseif cddr $(l) \simeq$ **nil** **then** list (`'plus`, car (l), cadr (l))
else list (`'plus`, car (l), plus-tree (cdr (l))) **endif**

THEOREM: numberp-eval\$-plus
$(\text{car}(x) = \texttt{'plus}) \rightarrow (\text{eval\$}(\mathbf{t}, x, a) \in \mathbf{N})$

THEOREM: numberp-eval\$-plus-tree
$\text{eval\$}(\mathbf{t}, \text{plus-tree}(l), a) \in \mathbf{N}$

THEOREM: member-implies-plus-tree-greatereqp
$(x \in y) \rightarrow (\text{eval\$}(\mathbf{t}, \text{plus-tree}(y), a) \not< \text{eval\$}(\mathbf{t}, x, a))$

THEOREM: plus-tree-delete
$\text{eval\$}(\mathbf{t}, \text{plus-tree}(\text{delete}(x, y)), a)$
$= \ \ \textbf{if } x \in y \textbf{ then } \text{eval\$}(\mathbf{t}, \text{plus-tree}(y), a) - \text{eval\$}(\mathbf{t}, x, a)$
$\quad\ \textbf{else } \text{eval\$}(\mathbf{t}, \text{plus-tree}(y), a) \textbf{ endif}$

THEOREM: subbagp-implies-plus-tree-geq
$\text{subbagp}(x, y) \rightarrow (\text{eval\$}(\mathbf{t}, \text{plus-tree}(y), a) \not< \text{eval\$}(\mathbf{t}, \text{plus-tree}(x), a))$

THEOREM: plus-tree-bagdiff
$\text{subbagp}(x, y)$
$\rightarrow \ \ (\text{eval\$}(\mathbf{t}, \text{plus-tree}(\text{bagdiff}(y, x)), a)$
$\qquad = \ \ (\text{eval\$}(\mathbf{t}, \text{plus-tree}(y), a) - \text{eval\$}(\mathbf{t}, \text{plus-tree}(x), a)))$

THEOREM: numberp-eval\$-bridge
$(\text{eval\$}(\mathbf{t}, z, a) = \text{eval\$}(\mathbf{t}, \text{plus-tree}(x), a)) \rightarrow (\text{eval\$}(\mathbf{t}, z, a) \in \mathbf{N})$

THEOREM: bridge-to-subbagp-implies-plus-tree-geq
$(\text{subbagp}(y, \text{plus-fringe}(z))$
$\ \wedge \ \ (\text{eval\$}(\mathbf{t}, z, a) = \text{eval\$}(\mathbf{t}, \text{plus-tree}(\text{plus-fringe}(z)), a)))$
$\rightarrow \ \ ((\text{eval\$}(\mathbf{t}, z, a) < \text{eval\$}(\mathbf{t}, \text{plus-tree}(y), a)) = \mathbf{f})$

THEOREM: eval\$-plus-tree-append
$\text{eval\$}(\mathbf{t}, \text{plus-tree}(\text{append}(x, y)), a)$
$= \ \ (\text{eval\$}(\mathbf{t}, \text{plus-tree}(x), a) + \text{eval\$}(\mathbf{t}, \text{plus-tree}(y), a))$

THEOREM: plus-tree-plus-fringe
$\text{eval\$}(\mathbf{t}, \text{plus-tree}(\text{plus-fringe}(x)), a) = \text{fix}(\text{eval\$}(\mathbf{t}, x, a))$

THEOREM: member-implies-numberp
$((c \in \text{plus-fringe}(x)) \wedge (\text{eval\$}(\mathbf{t}, c, a) \in \mathbf{N})) \rightarrow (\text{eval\$}(\mathbf{t}, x, a) \in \mathbf{N})$

THEOREM: cadr-eval\$-list
$(\text{car}(\text{eval\$}(\texttt{'list}, x, a)) = \text{eval\$}(\mathbf{t}, \text{car}(x), a))$
$\wedge \ \ (\text{cdr}(\text{eval\$}(\texttt{'list}, x, a))$
$\qquad = \ \ \textbf{if } \text{listp}(x) \textbf{ then } \text{eval\$}(\texttt{'list}, \text{cdr}(x), a)$
$\qquad\quad\ \ \textbf{else } \texttt{0} \textbf{ endif})$

THEOREM: eval$-quote
eval$ (**t**, cons (`'quote`, *args*), *a*) = car (*args*)

THEOREM: listp-eval$
listp (eval$ (`'list`, *x*, *a*)) = listp (*x*)

```
; the meta lemma to cancel identical plus terms in equality.  For example,
;   (EQUAL (PLUS A B C) (PLUS B D E)) => (EQUAL (PLUS A C) (PLUS D E)).
```

DEFINITION:

```
cancel-equal-plus (x)
=   if listp (x) ∧ (car (x) = 'equal)
    then if listp (cadr (x))
              ∧   (caadr (x) = 'plus)
              ∧   listp (caddr (x))
              ∧   (caaddr (x) = 'plus)
         then list ('equal,
                    plus-tree (bagdiff (plus-fringe (cadr (x)),
                                        bagint (plus-fringe (cadr (x)),
                                                plus-fringe (caddr (x))))),
                    plus-tree (bagdiff (plus-fringe (caddr (x)),
                                        bagint (plus-fringe (cadr (x)),
                                                plus-fringe (caddr (x))))))
         elseif listp (cadr (x))
                ∧   (caadr (x) = 'plus)
                ∧   (caddr (x) ∈ plus-fringe (cadr (x)))
         then list ('if,
                    list ('numberp, caddr (x)),
                    list ('equal,
                          plus-tree (delete (caddr (x), plus-fringe (cadr (x)))),
                          ''0),
                    list ('quote, f))
         elseif listp (caddr (x))
                ∧   (caaddr (x) = 'plus)
                ∧   (cadr (x) ∈ plus-fringe (caddr (x)))
         then list ('if,
                    list ('numberp, cadr (x)),
                    list ('equal,
                          ''0,
                          plus-tree (delete (cadr (x), plus-fringe (caddr (x))))),
                    list ('quote, f))
         else x endif
    else x endif
```

THEOREM: correctness-of-cancel-equal-plus
eval$ (**t**, *x*, *a*) = eval$ (**t**, cancel-equal-plus (*x*), *a*)

```
; the meta lemma to cancel identical plus terms in lessp.  For example,
; (DIFFERENCE (PLUS A B C) (PLUS B D E)) => (DIFFERENCE (PLUS A C) (PLUS D E)).
```

Definition:
cancel-difference-plus (x)

```
=   if listp (x) ∧ (car (x) = 'difference)
    then if listp (cadr (x))
              ∧  (caadr (x) = 'plus)
              ∧  listp (caddr (x))
              ∧  (caaddr (x) = 'plus)
         then list ('difference,
                    plus-tree (bagdiff (plus-fringe (cadr (x)),
                                        bagint (plus-fringe (cadr (x)),
                                                plus-fringe (caddr (x))))),
                    plus-tree (bagdiff (plus-fringe (caddr (x)),
                                        bagint (plus-fringe (cadr (x)),
                                                plus-fringe (caddr (x))))))
         elseif listp (cadr (x))
                  ∧  (caadr (x) = 'plus)
                  ∧  (caddr (x) ∈ plus-fringe (cadr (x)))
         then plus-tree (delete (caddr (x), plus-fringe (cadr (x))))
         elseif listp (caddr (x))
                  ∧  (caaddr (x) = 'plus)
                  ∧  (cadr (x) ∈ plus-fringe (caddr (x)))  then ''0
         else x endif
    else x endif
```

Theorem: correctness-of-cancel-difference-plus
eval\$ (**t**, x, a) = eval\$ (**t**, cancel-difference-plus (x), a)

```
; the meta lemma to cancel identical plus terms in lessp.  For example,
;   (LESSP (PLUS A B C) (PLUS B D E)) => (LESSP (PLUS A C) (PLUS D E)).
```

Definition:
cancel-lessp-plus (x)

```
=   if listp (x) ∧ (car (x) = 'lessp)
    then if listp (cadr (x))
              ∧  (caadr (x) = 'plus)
              ∧  listp (caddr (x))
              ∧  (caaddr (x) = 'plus)
         then list ('lessp,
                    plus-tree (bagdiff (plus-fringe (cadr (x)),
                                        bagint (plus-fringe (cadr (x)),
                                                plus-fringe (caddr (x))))),
                    plus-tree (bagdiff (plus-fringe (caddr (x)),
```

$\text{bagint}\,(\text{plus-fringe}\,(\text{cadr}\,(x)),$
$\quad\text{plus-fringe}\,(\text{caddr}\,(x))))))$

elseif $\text{listp}\,(\text{cadr}\,(x))$
$\quad\land\quad (\text{caadr}\,(x) = \texttt{'plus})$
$\quad\land\quad (\text{caddr}\,(x) \in \text{plus-fringe}\,(\text{cadr}\,(x)))$
then $\text{list}\,(\texttt{'quote}, \mathbf{f})$
elseif $\text{listp}\,(\text{caddr}\,(x))$
$\quad\land\quad (\text{caaddr}\,(x) = \texttt{'plus})$
$\quad\land\quad (\text{cadr}\,(x) \in \text{plus-fringe}\,(\text{caddr}\,(x)))$
then $\text{list}\,(\texttt{'not},$
$\quad\text{list}\,(\texttt{'zerop},$
$\quad\quad\text{plus-tree}\,(\text{delete}\,(\text{cadr}\,(x),\ \text{plus-fringe}\,(\text{caddr}\,(x))))))$
else x **endif**
else x **endif**

THEOREM: correctness-of-cancel-lessp-plus
$\text{eval\$}\,(\mathbf{t},\ x,\ a) = \text{eval\$}\,(\mathbf{t},\ \text{cancel-lessp-plus}\,(x),\ a)$

THEOREM: plus-lessp-cancel-2
$((y + x) < (x + z)) = (y < z)$

`; lemmas about times.`

THEOREM: times-zero
$((x \simeq \mathtt{0}) \lor (y \simeq \mathtt{0})) \rightarrow ((x * y) = \mathtt{0})$

THEOREM: times-distributes-plus
$(x * (y + z)) = ((x * y) + (x * z))$

THEOREM: times-add1
$(x * (1 + y)) = (x + (x * y))$

THEOREM: times-commutativity
$(x * z) = (z * x)$

THEOREM: times-commutativity1
$(x * (y * z)) = (y * (x * z))$

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

THEOREM: times-equal-1
$((x * y) = \mathtt{1}) = ((x = \mathtt{1}) \land (y = \mathtt{1}))$

THEOREM: times-1
$(\mathtt{1} * x) = \text{fix}\,(x)$

THEOREM: times-add1-sub1
$(1 + (a * (b - 1)))$
$= \ \textbf{if } (a \simeq 0) \vee (b \simeq 0) \textbf{ then } 1$
$\quad \textbf{else } (a * b) - (a - 1) \textbf{ endif}$

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

```
; x is a boolean value, iff x is either T or F.
```

DEFINITION: $\text{boolean}(x) = (\text{truep}(x) \vee \text{falsep}(x))$

THEOREM: equal-iff
$(\text{boolean}(p) \wedge \text{boolean}(q)) \rightarrow ((p = q) = (p \leftrightarrow q))$

THEOREM: times-equal-cancel0
$(((x * y) = y)$
$\quad = ((y \in \mathbf{N})$
$\qquad \wedge \ \textbf{if } y = 0 \textbf{ then t}$
$\qquad\quad \textbf{else } x = 1 \textbf{ endif}))$
$\wedge \ (((y * x) = y)$
$\qquad = ((y \in \mathbf{N})$
$\qquad\quad \wedge \ \textbf{if } y = 0 \textbf{ then t}$
$\qquad\qquad \textbf{else } x = 1 \textbf{ endif}))$

THEOREM: times-equal-cancel
$((x * y) = (x * z)) = ((x \simeq 0) \vee (\text{fix}(y) = \text{fix}(z)))$

THEOREM: times-lessp-0
$(0 < (x * y)) = ((0 < x) \wedge (0 < y))$

THEOREM: times-lessp-1
$(1 < (x * y))$
$= \ ((x \not\simeq 0) \wedge (y \not\simeq 0) \wedge (\neg ((x = 1) \wedge (y = 1))))$

THEOREM: times-lessp-cancel0
$(((x * y) < x) = ((x \not\simeq 0) \wedge (y \simeq 0)))$
$\wedge \ ((x < (x * y)) = ((x \not\simeq 0) \wedge (y \not\simeq 0) \wedge (y \neq 1)))$

THEOREM: times-lessp-cancel
$((x * y) < (x * z)) = ((x \not\simeq 0) \wedge (y < z))$

EVENT: Disable equal-iff.

THEOREM: times-lessp-cancel-1
$((x * y) < (x + (x * z))) = ((x \not\simeq 0) \wedge (y \leq z))$

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

THEOREM: times-distributes-difference
$((x * y) - (x * z)) = (x * (y - z))$

THEOREM: times-distributes-difference1
$((y * x) - (z * x)) = (x * (y - z))$

THEOREM: times2-add1-lessp-cancel
$((1 + (2 * i)) < (2 * j)) = (i < j)$

```
; meta lemmas for times.  Stolen and modified from naturals.events.
```

DEFINITION:
times-fringe (x)
$=$ **if** listp $(x) \wedge$ (car $(x) =$ `'times`)
then append (times-fringe (cadr (x)), times-fringe (caddr (x)))
else cons $(x$, **nil**) **endif**

DEFINITION:
times-tree (x)
$=$ **if** $x \simeq$ **nil** **then** `''1`
elseif cdr $(x) \simeq$ **nil** **then** list (`'fix`, car (x))
elseif cddr $(x) \simeq$ **nil** **then** list (`'times`, car (x), cadr (x))
else list (`'times`, car (x), times-tree (cdr (x))) **endif**

DEFINITION:
and-not-zerop-tree (x)
$=$ **if** $x \simeq$ **nil** **then** `'(true)`
elseif cdr $(x) \simeq$ **nil** **then** list (`'not`, list (`'zerop`, car (x)))
else list (`'and`,
list (`'not`, list (`'zerop`, car (x))),
and-not-zerop-tree (cdr (x))) **endif**

THEOREM: numberp-eval\$-times
(car $(x) =$ `'times`) $\rightarrow$ (eval\$ ($\mathbf{t}$, x, a) $\in \mathbf{N}$)

THEOREM: eval\$-times-tree-numberp
eval\$ ($\mathbf{t}$, times-tree (x), a) $\in \mathbf{N}$

THEOREM: eval\$-times-member
$(e \in x)$
$\rightarrow$ (eval\$ ($\mathbf{t}$, times-tree (x), a)
$=$ (eval\$ ($\mathbf{t}$, e, a) $*$ eval\$ ($\mathbf{t}$, times-tree (delete (e, x)), a)))

THEOREM: zerop-makes-times-tree-zero
$((\neg\ \text{eval\$}\,(\mathbf{t},\ \text{and-not-zerop-tree}\,(x),\ a)) \land \text{subbagp}\,(x,\ y))$
$\rightarrow\quad (\text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(y),\ a) = \mathtt{0})$

THEOREM: eval$-times-tree-append
$\text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(\text{append}\,(x,\ y)),\ a)$
$=\quad (\text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(x),\ a) * \text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(y),\ a))$

THEOREM: times-tree-times-fringe
$\text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(\text{times-fringe}\,(x)),\ a) = \text{fix}\,(\text{eval\$}\,(\mathbf{t},\ x,\ a))$

THEOREM: eval$-lessp-times-tree-bagdiff
$(\text{eval\$}\,(\mathbf{t},\ \text{and-not-zerop-tree}\,(x),\ a) \land \text{subbagp}\,(x,\ y) \land \text{subbagp}\,(x,\ z))$
$\rightarrow\quad ((\text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(\text{bagdiff}\,(y,\ x)),\ a)$
$\qquad <\quad \text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(\text{bagdiff}\,(z,\ x)),\ a))$
$\qquad =\quad (\text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(y),\ a) < \text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(z),\ a)))$

THEOREM: zerop-makes-lessp-false-bridge
$(\neg\ \text{eval\$}\,(\mathbf{t},$
$\qquad\qquad \text{and-not-zerop-tree}\,(\text{bagint}\,(\text{times-fringe}\,(\text{cons}\,(\texttt{'times},\ x)),$
$\qquad\qquad\qquad\qquad\qquad \text{times-fringe}\,(\text{cons}\,(\texttt{'times},\ y)))),$
$\qquad\qquad a))$
$\rightarrow\quad (((\text{eval\$}\,(\mathbf{t},\ \text{car}\,(x),\ a) * \text{eval\$}\,(\mathbf{t},\ \text{cadr}\,(x),\ a))$
$\qquad <\quad (\text{eval\$}\,(\mathbf{t},\ \text{car}\,(y),\ a) * \text{eval\$}\,(\mathbf{t},\ \text{cadr}\,(y),\ a)))$
$\qquad =\quad \mathbf{f})$

THEOREM: and-not-zerop-tree-lessp
$\text{eval\$}\,(\mathbf{t},\ \text{and-not-zerop-tree}\,(x),\ a) = (\text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(x),\ a) \not< \mathtt{1})$

DEFINITION:
$\text{eval\$-and-not-zerop-tree-end}\,(w,\ x,\ a)$
$=\quad \text{eval\$}\,(\mathbf{t},\ \text{and-not-zerop-tree}\,(\text{delete}\,(w,\ x)),\ a)$

THEOREM: and-not-zerop-tree-delete
$(w \in x)$
$\rightarrow\quad (\text{eval\$}\,(\mathbf{t},\ \text{and-not-zerop-tree}\,(\text{delete}\,(w,\ x)),\ a)$
$\qquad =\quad \textbf{if}\ \text{eval\$}\,(\mathbf{t},\ w,\ a) \simeq \mathtt{0}\ \textbf{then}\ \text{eval\$-and-not-zerop-tree-end}\,(w,\ x,\ a)$
$\qquad\qquad \textbf{else}\ \text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(x),\ a) \not< \text{eval\$}\,(\mathbf{t},\ w,\ a)\ \textbf{endif})$

EVENT: Disable and-not-zerop-tree-lessp.

DEFINITION:
$\text{lessp-1-times-tree-delete-end}\,(w,\ x,\ a)$
$=\quad (\mathtt{1} < \text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(\text{delete}\,(w,\ x)),\ a))$

THEOREM: lessp-1-times-tree-delete
```
 (w ∈ x)
 →   ((1 < eval$ (t, times-tree (delete (w, x)), a))
       =   if eval$ (t, w, a) ≃ 0  then lessp-1-times-tree-delete-end (w, x, a)
           else eval$ (t, w, a) < eval$ (t, times-tree (x), a) endif)
```

THEOREM: eval$-times-fringe-member-zero
```
 ((e ∈ times-fringe (cons ('times, x))) ∧ (eval$ (t, e, a) ≃ 0))
 →   ((eval$ (t, car (x), a) ∗ eval$ (t, cadr (x), a)) = 0)
```

DEFINITION:
```
cancel-lessp-times (x)
=   if listp (x) ∧ (car (x) = 'lessp)
    then if (caadr (x) = 'times) ∧ (caaddr (x) = 'times)
         then if listp (bagint (times-fringe (cadr (x)),
                                times-fringe (caddr (x))))
              then list ('and,
                         and-not-zerop-tree (bagint (times-fringe (cadr (x)),
                                                     times-fringe (caddr (x)))),
                         list ('lessp,
                               times-tree (bagdiff (times-fringe (cadr (x)),
                                                    bagint (times-fringe (cadr (x)),
                                                            times-fringe (caddr (x))))),
                               times-tree (bagdiff (times-fringe (caddr (x)),
                                                    bagint (times-fringe (cadr (x)),
                                                            times-fringe (caddr (x)))))))
              else x endif
         elseif listp (cadr (x))
                ∧   (caadr (x) = 'times)
                ∧   (caddr (x) ∈ times-fringe (cadr (x)))
         then list ('and,
                    list ('not, list ('zerop, caddr (x))),
                    list ('not,
                          and-not-zerop-tree (delete (caddr (x),
                                                      times-fringe (cadr (x))))))
         elseif listp (caddr (x))
                ∧   (caaddr (x) = 'times)
                ∧   (cadr (x) ∈ times-fringe (caddr (x)))
         then list ('and,
                    list ('not, list ('zerop, cadr (x))),
                    list ('lessp,
                          ''1,
                          times-tree (delete (cadr (x), times-fringe (caddr (x))))))
         else x endif
    else x endif
```

```
; the meta lemma to cancel identical times terms in lessp. For example,
; (lessp (times b (times c d)) (times b d)) =>
;         (and (and (not (zerop b)) (not (zerop d))) (lessp (fix c) 1))
```

THEOREM: correctness-of-cancel-lessp-times
eval$ (**t**, x, a) = eval$ (**t**, cancel-lessp-times (x), a)

EVENT: Disable and-not-zerop-tree-delete.

EVENT: Disable lessp-1-times-tree-delete.

THEOREM: eval$-equal-times-tree-bagdiff
(eval$ (**t**, and-not-zerop-tree (x), a) $\wedge$ subbagp (x, y) $\wedge$ subbagp (x, z))
$\rightarrow$ ((eval$ (**t**, times-tree (bagdiff (y, x)), a)
  = eval$ (**t**, times-tree (bagdiff (z, x)), a))
 = (eval$ (**t**, times-tree (y), a) = eval$ (**t**, times-tree (z), a)))

THEOREM: zerop-makes-equal-true-bridge
($\neg$ eval$ (**t**,
  and-not-zerop-tree (bagint (times-fringe (cons (`'times`, x)),
    times-fringe (cons (`'times`, y)))),
  a))
$\rightarrow$ (((eval$ (**t**, car (x), a) $*$ eval$ (**t**, cadr (x), a))
  = (eval$ (**t**, car (y), a) $*$ eval$ (**t**, cadr (y), a)))
 = **t**)

DEFINITION:
equal-1-eval$-times-tree-delete-end ($w$, x, a)
= (eval$ (**t**, times-tree (delete (w, x)), a) = `1`)

THEOREM: equal-1-times-tree-delete
($w \in x$)
$\rightarrow$ ((eval$ (**t**, times-tree (delete (w, x)), a) = `1`)
 = **if** eval$ (**t**, w, a) $\simeq$ `0`
  then equal-1-eval$-times-tree-delete-end ($w$, x, a)
  else eval$ (**t**, times-tree (x), a) = eval$ (**t**, w, a) **endif**)

DEFINITION:
cancel-equal-times (x)
= **if** car (x) = `'equal`
 then if (caadr (x) = `'times`) $\wedge$ (caaddr (x) = `'times`)
  then if listp (bagint (times-fringe (cadr (x)),
    times-fringe (caddr (x))))
   then list (`'if`,

```
                          and-not-zerop-tree (bagint (times-fringe (cadr (x)),
                                                      times-fringe (caddr (x)))),
                          list ('equal,
                                times-tree (bagdiff (times-fringe (cadr (x)),
                                                     bagint (times-fringe (cadr (x)),
                                                             times-fringe (caddr (x))))),
                                times-tree (bagdiff (times-fringe (caddr (x)),
                                                     bagint (times-fringe (cadr (x)),
                                                             times-fringe (caddr (x)))))),
                          '(true))
                else x endif
        elseif listp (cadr (x))
                ∧   (caadr (x) = 'times)
                ∧   (caddr (x) ∈ times-fringe (cadr (x)))
        then list ('and,
                   list ('numberp, caddr (x)),
                   list ('or,
                         list ('equal, caddr (x), ''0),
                         list ('equal,
                               times-tree (delete (caddr (x),
                                                   times-fringe (cadr (x)))),
                               ''1)))
        elseif listp (caddr (x))
                ∧   (caaddr (x) = 'times)
                ∧   (cadr (x) ∈ times-fringe (caddr (x)))
        then list ('and,
                   list ('numberp, cadr (x)),
                   list ('or,
                         list ('equal, cadr (x), ''0),
                         list ('equal,
                               times-tree (delete (cadr (x),
                                                   times-fringe (caddr (x)))),
                               ''1)))
        else x endif
    else x endif
```

```
; the meta lemma to cancel identical times term in equality.  For example,
;     (equal (times b (times c d)) (times b d)) =>
;          (or (or (zerop b) (zerop d)) (equal (fix c) 1))
```

THEOREM: correctness-of-cancel-equal-times
eval$ (**t**, x, a) = eval$ (**t**, cancel-equal-times (x), a)

EVENT: For efficiency, compile those definitions not yet compiled.

```
; lemmas about exp.
```

DEFINITION:
$\exp(x, y)$
$= \ \textbf{if } y \simeq \mathtt{0} \textbf{ then } \mathtt{1}$
$\quad \textbf{else } x * \exp(x, y - 1) \textbf{ endif}$

THEOREM: exp-of-0
$\exp(\mathtt{0}, k)$
$= \ \textbf{if } k \simeq \mathtt{0} \textbf{ then } \mathtt{1}$
$\quad \textbf{else } \mathtt{0} \textbf{ endif}$

THEOREM: exp-of-1
$\exp(\mathtt{1}, k) = \mathtt{1}$

THEOREM: exp-plus
$(\exp(x, y) * \exp(x, z)) = \exp(x, y + z)$

THEOREM: exp-times
$\exp(x * y, z) = (\exp(x, z) * \exp(y, z))$

THEOREM: exp-exp
$\exp(\exp(x, y), z) = \exp(x, y * z)$

THEOREM: exp-of-2-0
$(m \not\simeq \mathtt{0}) \rightarrow (\exp(m, n) \not< \mathtt{1})$

THEOREM: exp-of-2-1
$(\mathtt{1} < \exp(\mathtt{2}, n)) = (n \not\simeq \mathtt{0})$

THEOREM: exp-lessp
$(\exp(x, y) < \exp(x, z))$
$= \ \textbf{if } x \simeq \mathtt{0} \textbf{ then } (y \not\simeq \mathtt{0}) \wedge (z \simeq \mathtt{0})$
$\quad \textbf{elseif } x = \mathtt{1} \textbf{ then } \mathbf{f}$
$\quad \textbf{else } y < z \textbf{ endif}$

EVENT: Disable times.

THEOREM: times-exp2-lessp
$((i * \exp(\mathtt{2}, j)) < \exp(\mathtt{2}, k)) = (i < \exp(\mathtt{2}, k - j))$

```
; lemmas about remainder and quotient.
```

THEOREM: remainder-exit
$(i < j) \rightarrow ((i \textbf{ mod } j) = \text{fix}(i))$

THEOREM: quotient-exit
$(i < j) \rightarrow ((i \div j) = \mathtt{0})$

THEOREM: remainder-0
$((\mathtt{0} \ \mathbf{mod} \ x) = \mathtt{0}) \wedge ((x \ \mathbf{mod} \ \mathtt{0}) = \mathrm{fix}\,(x))$

THEOREM: quotient-0
$((\mathtt{0} \div x) = \mathtt{0}) \wedge ((x \div \mathtt{0}) = \mathtt{0})$

THEOREM: remainder-1
$$\begin{array}{ll} ((\mathtt{1} \ \mathbf{mod} \ x) & \\ \ \ = & \mathbf{if} \ x = \mathtt{1} \ \ \mathbf{then} \ \mathtt{0} \\ & \mathbf{else} \ \mathtt{1} \ \mathbf{endif}) \\ \wedge & ((x \ \mathbf{mod} \ \mathtt{1}) = \mathtt{0}) \end{array}$$

THEOREM: quotient-1
$$\begin{array}{ll} ((\mathtt{1} \div x) & \\ \ \ = & \mathbf{if} \ x = \mathtt{1} \ \ \mathbf{then} \ \mathtt{1} \\ & \mathbf{else} \ \mathtt{0} \ \mathbf{endif}) \\ \wedge & ((m \div \mathtt{1}) = \mathrm{fix}\,(m)) \end{array}$$

THEOREM: remainder-x-x
$(x \ \mathbf{mod} \ x) = \mathtt{0}$

THEOREM: quotient-x-x
$$\begin{array}{ll} (x \div x) & \\ = & \mathbf{if} \ x \simeq \mathtt{0} \ \ \mathbf{then} \ \mathtt{0} \\ & \mathbf{else} \ \mathtt{1} \ \mathbf{endif} \end{array}$$

THEOREM: quotient-equal-0
$((m \div n) = \mathtt{0}) = ((m \simeq \mathtt{0}) \vee (n \simeq \mathtt{0}) \vee (m < n))$

THEOREM: remainder-2x
$((x + x) \ \mathbf{mod} \ 2) = \mathtt{0}$

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

THEOREM: remainder-2x-add1
$((x + (1 + x)) \ \mathbf{mod} \ 2) = \mathtt{1}$

THEOREM: quotient-2x-add1
$((x + (1 + x)) \div 2) = \mathrm{fix}\,(x)$

```
; A generalization lemma about quotient.
```

THEOREM: quotient-generalize
$((m \div n) = 0)$
$= \quad \textbf{if } (m \simeq 0) \lor (n \simeq 0) \textbf{ then t}$
$\quad \textbf{else } m < n \textbf{ endif}$

EVENT: Disable quotient-generalize.

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

THEOREM: quotient-lessp
$((m \div n) < m) = ((m \not\simeq 0) \land ((n \simeq 0) \lor (n \neq 1)))$

THEOREM: remainder-lessp-linear
$(y \not\simeq 0) \rightarrow ((x \textbf{ mod } y) < y)$

THEOREM: quotient-lessp-linear
$((x \not\simeq 0) \land (1 < y)) \rightarrow ((x \div y) < x)$

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

THEOREM: remainder-wrt-2
$(n \textbf{ mod } 2) < 2$

THEOREM: remainder-plus1
$((i \textbf{ mod } j) = 0) \rightarrow (((x + i) \textbf{ mod } j) = (x \textbf{ mod } j))$

THEOREM: remainder-plus2
$((i \textbf{ mod } j) = 0) \rightarrow (((i + x) \textbf{ mod } j) = (x \textbf{ mod } j))$

THEOREM: remainder-times
$(((x * y) \textbf{ mod } y) = 0) \land (((y * x) \textbf{ mod } y) = 0)$

THEOREM: remainder-plus-times1
$((x + (y * z)) \textbf{ mod } y) = (x \textbf{ mod } y)$

THEOREM: remainder-plus-times2
$((x + (z * y)) \textbf{ mod } y) = (x \textbf{ mod } y)$

THEOREM: remainder-plus-plus
$((i \textbf{ mod } j) = 0) \rightarrow (((x + y + i) \textbf{ mod } j) = ((x + y) \textbf{ mod } j))$

THEOREM: remainder-plus-add1
$((i \textbf{ mod } j) = 0) \rightarrow (((1 + (x + i)) \textbf{ mod } j) = ((1 + x) \textbf{ mod } j))$

THEOREM: remainder-plus-difference1
$$((x \not< z) \land ((y \textbf{ mod } w) = \mathtt{0})) \rightarrow ((((x + y) - z) \textbf{ mod } w) = ((x - z) \textbf{ mod } w))$$

THEOREM: remainder-plus-difference2
$$((y \not< z) \land ((x \textbf{ mod } w) = \mathtt{0})) \rightarrow ((((x + y) - z) \textbf{ mod } w) = ((y - z) \textbf{ mod } w))$$

THEOREM: remainder-plus-plus-times1
$$((x + w + (y * z)) \textbf{ mod } y) = ((x + w) \textbf{ mod } y)$$

THEOREM: remainder-plus-plus-times2
$$((x + w + (z * y)) \textbf{ mod } y) = ((x + w) \textbf{ mod } y)$$

THEOREM: remainder-difference
$$((y \textbf{ mod } z) = \mathtt{0}) \rightarrow (((x - y) \textbf{ mod } z) = \textbf{if } x < y \textbf{ then } \mathtt{0} \textbf{ else } x \textbf{ mod } z \textbf{ endif})$$

THEOREM: remainder-difference-times1
$$((x - (y * z)) \textbf{ mod } z) = \textbf{if } x < (y * z) \textbf{ then } \mathtt{0} \textbf{ else } x \textbf{ mod } z \textbf{ endif}$$

THEOREM: remainder-difference-times2
$$((x - (y * z)) \textbf{ mod } y) = \textbf{if } x < (y * z) \textbf{ then } \mathtt{0} \textbf{ else } x \textbf{ mod } y \textbf{ endif}$$

THEOREM: quotient-plus1
$$((i \textbf{ mod } j) = \mathtt{0}) \rightarrow (((x + i) \div j) = ((x \div j) + (i \div j)))$$

THEOREM: quotient-plus2
$$((i \textbf{ mod } j) = \mathtt{0}) \rightarrow (((i + x) \div j) = ((i \div j) + (x \div j)))$$

THEOREM: quotient-times
$$(((x * y) \div y) = \textbf{if } y \simeq \mathtt{0} \textbf{ then } \mathtt{0} \textbf{ else } \mathrm{fix}\,(x) \textbf{ endif}) \land (((y * x) \div y) = \textbf{if } y \simeq \mathtt{0} \textbf{ then } \mathtt{0} \textbf{ else } \mathrm{fix}\,(x) \textbf{ endif})$$

THEOREM: quotient-plus-times1

$$((x + (y * z)) \div y) \\ = \quad ((x \div y) \\ \quad + \quad \textbf{if } y \simeq \mathtt{0} \textbf{ then } \mathtt{0} \\ \quad\quad \textbf{else } \mathrm{fix}\,(z) \textbf{ endif})$$

THEOREM: quotient-plus-times2

$$((x + (z * y)) \div y) \\ = \quad ((x \div y) \\ \quad + \quad \textbf{if } y \simeq \mathtt{0} \textbf{ then } \mathtt{0} \\ \quad\quad \textbf{else } \mathrm{fix}\,(z) \textbf{ endif})$$

THEOREM: quotient-plus-plus

$$((i \textbf{ mod } j) = \mathtt{0}) \\ \rightarrow \quad (((x + y + i) \div j) = (((x + y) \div j) + (i \div j)))$$

THEOREM: quotient-plus-add1

$$((i \textbf{ mod } j) = \mathtt{0}) \\ \rightarrow \quad (((1 + (x + i)) \div j) = (((1 + x) \div j) + (i \div j)))$$

THEOREM: quotient-difference-plus1

$$((x \not< z) \wedge ((y \textbf{ mod } w) = \mathtt{0})) \\ \rightarrow \quad ((((x + y) - z) \div w) = (((x - z) \div w) + (y \div w)))$$

THEOREM: quotient-difference-plus2

$$((y \not< z) \wedge ((x \textbf{ mod } w) = \mathtt{0})) \\ \rightarrow \quad ((((x + y) - z) \div w) = (((y - z) \div w) + (x \div w)))$$

THEOREM: quotient-difference

$$((y \textbf{ mod } z) = \mathtt{0}) \\ \rightarrow \quad (((x - y) \div z) \\ \quad = \quad \textbf{if } x < y \textbf{ then } \mathtt{0} \\ \quad\quad \textbf{else } (x \div z) - (y \div z) \textbf{ endif})$$

THEOREM: quotient-difference-times1

$$((x - (y * z)) \div z) \\ = \quad \textbf{if } x < (y * z) \textbf{ then } \mathtt{0} \\ \quad \textbf{else } (x \div z) - \mathrm{fix}\,(y) \textbf{ endif}$$

THEOREM: quotient-difference-times2

$$((x - (y * z)) \div y) \\ = \quad \textbf{if } x < (y * z) \textbf{ then } \mathtt{0} \\ \quad \textbf{else } (x \div y) - \mathrm{fix}\,(z) \textbf{ endif}$$

EVENT: Disable remainder-difference.

EVENT: Disable quotient-difference.

THEOREM: remainder-sub1

$$
\begin{aligned}
&((m - 1) \textbf{ mod } n)\\
= \quad & \textbf{if } n \simeq \texttt{0} \textbf{ then } m - 1\\
& \textbf{elseif } (m \textbf{ mod } n) = \texttt{0}\\
& \textbf{then if } m \simeq \texttt{0} \textbf{ then } \texttt{0}\\
& \qquad \textbf{else } n - 1 \textbf{ endif}\\
& \textbf{else } (m \textbf{ mod } n) - 1 \textbf{ endif}
\end{aligned}
$$

THEOREM: quotient-sub1

$$
\begin{aligned}
&((m - 1) \div n)\\
= \quad & \textbf{if } (m \textbf{ mod } n) = \texttt{0} \textbf{ then } (m \div n) - 1\\
& \textbf{else } m \div n \textbf{ endif}
\end{aligned}
$$

THEOREM: remainder-quotient

$$((y * (x \div y)) + (x \textbf{ mod } y)) = \mathrm{fix}\,(x)$$

THEOREM: remainder-quotient-elim

$$((y \not\simeq \texttt{0}) \wedge (x \in \mathbf{N})) \rightarrow (((x \textbf{ mod } y) + (y * (x \div y))) = x)$$

THEOREM: remainder-add1

$$
\begin{aligned}
&((1 + m) \textbf{ mod } n)\\
= \quad & \textbf{if } n \simeq \texttt{0} \textbf{ then } 1 + m\\
& \textbf{elseif } (m \textbf{ mod } n) = (n - 1) \textbf{ then } \texttt{0}\\
& \textbf{else } 1 + (m \textbf{ mod } n) \textbf{ endif}
\end{aligned}
$$

THEOREM: quotient-add1

$$
\begin{aligned}
&((1 + m) \div n)\\
= \quad & \textbf{if } n \simeq \texttt{0} \textbf{ then } \texttt{0}\\
& \textbf{elseif } (m \textbf{ mod } n) = (n - 1) \textbf{ then } 1 + (m \div n)\\
& \textbf{else } m \div n \textbf{ endif}
\end{aligned}
$$

THEOREM: times-plus-lessp

$$(x < d) \rightarrow (((x + (b * d)) < (c * d)) = (b < c))$$

THEOREM: quotient-shrink-fast

$$x \not< (y * (x \div y))$$

THEOREM: remainder-plus-remainder1

$$((x + (y \textbf{ mod } z)) \textbf{ mod } z) = ((x + y) \textbf{ mod } z)$$

THEOREM: remainder-difference-remainder1

$$
\begin{aligned}
&\textbf{if } x < y \textbf{ then f}\\
&\textbf{else t endif}\\
\rightarrow \quad & (((x - (y \textbf{ mod } z)) \textbf{ mod } z) = ((x - y) \textbf{ mod } z))
\end{aligned}
$$

THEOREM: remainder-plus-remainder2

$$((x + y + (z \textbf{ mod } k)) \textbf{ mod } k) = ((x + y + z) \textbf{ mod } k)$$

THEOREM: remainder-plus-remainder
$(((x \textbf{ mod } z) + (y \textbf{ mod } z)) \textbf{ mod } z) = ((x + y) \textbf{ mod } z)$

THEOREM: remainder-crock
$(y < z) \rightarrow (((y * x) \textbf{ mod } (x * z)) = (y * x))$

THEOREM: times-distributes-remainder
$((x * y) \textbf{ mod } (x * z)) = (x * (y \textbf{ mod } z))$

THEOREM: quotient-crock
$(y < z) \rightarrow (((y * x) \div (x * z)) = \texttt{0})$

THEOREM: quotient-times-cancel
$((x * y) \div (x * z))$
$= \quad \textbf{if } x \simeq \texttt{0} \textbf{ then } \texttt{0}$
$\quad\quad \textbf{else } y \div z \textbf{ endif}$

EVENT: Disable remainder-crock.

EVENT: Disable quotient-crock.

THEOREM: remainder-distributes-times2-add1
$((1 + (\texttt{2} * y)) \textbf{ mod } (\texttt{2} * z)) = (1 + (\texttt{2} * (y \textbf{ mod } z)))$

THEOREM: quotient-distributes-times2-add1
$((1 + (\texttt{2} * y)) \div (\texttt{2} * z)) = (y \div z)$

THEOREM: quotient-exp
$(\texttt{1} < i)$
$\rightarrow \quad ((\exp(i, j) \div \exp(i, k))$
$\quad\quad = \quad \textbf{if } j < k \textbf{ then } \texttt{0}$
$\quad\quad\quad\quad \textbf{else } \exp(i, j - k) \textbf{ endif})$

THEOREM: remainder-exp
$(\texttt{1} < i)$
$\rightarrow \quad ((\exp(i, j) \textbf{ mod } \exp(i, k))$
$\quad\quad = \quad \textbf{if } j < k \textbf{ then } \exp(i, j)$
$\quad\quad\quad\quad \textbf{else } \texttt{0} \textbf{ endif})$

THEOREM: remainder-plus-cancel0
$(((i + j) \textbf{ mod } n) = i)$
$= \quad \textbf{if } n \simeq \texttt{0} \textbf{ then } (i \in \mathbf{N}) \land (j \simeq \texttt{0})$
$\quad\quad \textbf{elseif } i < n \textbf{ then } (i \in \mathbf{N}) \land ((j \textbf{ mod } n) = \texttt{0})$
$\quad\quad \textbf{else f endif}$

THEOREM: remainder-plus-cancel
$(((i + j) \textbf{ mod } n) = ((i + k) \textbf{ mod } n)) = ((j \textbf{ mod } n) = (k \textbf{ mod } n))$

THEOREM: quotient-times-lessp
$((x \div z) < y)$
$= \textbf{ if } z \simeq 0 \textbf{ then } y \not\simeq 0$
$\quad \textbf{else } x < (z * y) \textbf{ endif}$

THEOREM: quotient-quotient
$((x \div z) \div y) = (x \div (z * y))$

```
; we redefine the distribution law of times and plus, and disable the old one.
```

THEOREM: times-distributes-plus-new
$((x * y) + (x * z)) = (x * (y + z))$

EVENT: Disable times-distributes-plus.

```
; an induction hint for the next event.
```

DEFINITION:
quot2-sub12-induct (x, y, i, j)
$= \textbf{ if } i \simeq 0 \textbf{ then t}$
$\quad \textbf{else }$ quot2-sub12-induct $(x \div 2, y \div 2, i - 1, j - 1) \textbf{ endif}$

THEOREM: lessp-plus-times-exp2
$(x < \exp(2, i))$
$\rightarrow \quad (((x + (y * \exp(2, i))) < \exp(2, n))$
$\quad = \textbf{ if } y \simeq 0 \textbf{ then } x < \exp(2, n)$
$\quad\quad \textbf{else } y < \exp(2, n - i) \textbf{ endif})$

THEOREM: lessp-plus-exp2
$(x < \exp(2, i)) \rightarrow (((x + \exp(2, i)) < \exp(2, n)) = (i < n))$

THEOREM: remainder-times-exp2-1
$((x * \exp(2, i)) \textbf{ mod } \exp(2, j)) = ((x \textbf{ mod } \exp(2, j - i)) * \exp(2, i))$

THEOREM: remainder-times-exp2-2
$((\exp(2, i) * x) \textbf{ mod } \exp(2, j)) = ((x \textbf{ mod } \exp(2, j - i)) * \exp(2, i))$

```
; special cases of remainder-times-exp2.
```

THEOREM: remainder-times-exp2-3
$(((x * \exp(2, i)) \textbf{ mod } 2)$
$\quad = \textbf{ if } i \simeq 0 \textbf{ then } x \textbf{ mod } 2$
$\quad\quad \textbf{else } 0 \textbf{ endif})$
$\wedge \quad (((\exp(2, i) * x) \textbf{ mod } 2)$
$\quad = \textbf{ if } i \simeq 0 \textbf{ then } x \textbf{ mod } 2$
$\quad\quad \textbf{else } 0 \textbf{ endif})$

THEOREM: remainder-times-exp2-4

$$(((x * \exp(2, i) * y) \ \mathbf{mod}\ 2) = \mathbf{if}\ i \simeq 0\ \mathbf{then}\ (x * y)\ \mathbf{mod}\ 2\ \mathbf{else}\ 0\ \mathbf{endif}) \wedge (((x * y * \exp(2, i))\ \mathbf{mod}\ 2) = \mathbf{if}\ i \simeq 0\ \mathbf{then}\ (x * y)\ \mathbf{mod}\ 2\ \mathbf{else}\ 0\ \mathbf{endif})$$

THEOREM: quotient-times-exp2-1

$$((x * \exp(2, i)) \div \exp(2, j)) = \mathbf{if}\ i < j\ \mathbf{then}\ x \div \exp(2, j - i)\ \mathbf{else}\ x * \exp(2, i - j)\ \mathbf{endif}$$

THEOREM: quotient-times-exp2-2

$$((\exp(2, i) * x) \div \exp(2, j)) = \mathbf{if}\ i < j\ \mathbf{then}\ x \div \exp(2, j - i)\ \mathbf{else}\ x * \exp(2, i - j)\ \mathbf{endif}$$

THEOREM: quotient-times-exp2-3

$$(((x * \exp(2, i)) \div 2) = \mathbf{if}\ i \simeq 0\ \mathbf{then}\ x \div 2\ \mathbf{else}\ x * \exp(2, i - 1)\ \mathbf{endif}) \wedge (((\exp(2, i) * x) \div 2) = \mathbf{if}\ i \simeq 0\ \mathbf{then}\ x \div 2\ \mathbf{else}\ x * \exp(2, i - 1)\ \mathbf{endif})$$

THEOREM: quotient-times-exp2-4

$$(((x * \exp(2, i) * y) \div 2) = \mathbf{if}\ i \simeq 0\ \mathbf{then}\ (x * y) \div 2\ \mathbf{else}\ x * y * \exp(2, i - 1)\ \mathbf{endif}) \wedge (((x * y * \exp(2, i)) \div 2) = \mathbf{if}\ i \simeq 0\ \mathbf{then}\ (x * y) \div 2\ \mathbf{else}\ x * y * \exp(2, i - 1)\ \mathbf{endif})$$

THEOREM: remainder-remainder-exp2

$$((x\ \mathbf{mod}\ \exp(2, i))\ \mathbf{mod}\ \exp(2, j)) = \mathbf{if}\ i < j\ \mathbf{then}\ x\ \mathbf{mod}\ \exp(2, i)\ \mathbf{else}\ x\ \mathbf{mod}\ \exp(2, j)\ \mathbf{endif}$$

```
; a lemma for the event add-evenp.
```

THEOREM: remainder-remainder-2

$$((x\ \mathbf{mod}\ \exp(2, i))\ \mathbf{mod}\ 2) = \mathbf{if}\ i \simeq 0\ \mathbf{then}\ 0\ \mathbf{else}\ x\ \mathbf{mod}\ 2\ \mathbf{endif}$$

```
; logarithm is not used in the specification or the lemma library.  It
; comes in when we'd like to reason about the time complexity of programs.
```

THEOREM: times-lessp
$(x \leq z)$
$\rightarrow \quad ((x < (y * z))$
$\quad = \quad$ **if** $(y \simeq 0) \vee (z \simeq 0)$ **then f**
elseif $y = 1$ **then** $x < z$
else t endif)

DEFINITION:
$\log(b,\ x)$
$=$ **if** $(b \simeq 0) \vee (b = 1)$ **then** 0
elseif $x < b$ **then** 0
else $1 + \log(b,\ x \div b)$ **endif**

THEOREM: log-of-0
$\log(b,\ 0) = 0$

THEOREM: log-of-1
$(1 < b) \rightarrow (\log(b,\ 1) = 0)$

THEOREM: log-equal-0
$(\log(b,\ x) = 0) = ((b \simeq 0) \vee (b = 1) \vee (x < b))$

THEOREM: log-exp
$(1 < b) \rightarrow (\log(b,\ \exp(b,\ n)) = \mathrm{fix}(n))$

THEOREM: log-times-exp
$((1 < b) \wedge (x \not\simeq 0))$
$\rightarrow \quad ((\log(b,\ x * \exp(b,\ n)) = (n + \log(b,\ x)))$
$\quad \wedge \quad (\log(b,\ \exp(b,\ n) * x) = (n + \log(b,\ x))))$

THEOREM: log-times-exp-1
$((1 < b) \wedge (x \not\simeq 0))$
$\rightarrow \quad ((\log(b,\ x * b) = (1 + \log(b,\ x)))$
$\quad \wedge \quad (\log(b,\ b * x) = (1 + \log(b,\ x))))$

THEOREM: log-quotient-exp
$(1 < b) \rightarrow (\log(b,\ x \div \exp(b,\ i)) = (\log(b,\ x) - i))$

DEFINITION:
quotient2-induct $(b,\ x,\ y)$
$=$ **if** $(b \simeq 0) \vee (b = 1)$ **then** 0
elseif $(x \simeq 0) \vee (y \simeq 0)$ **then** 0
else quotient2-induct $(b,\ x \div b,\ y \div b)$ **endif**

THEOREM: log-leq
$(x \leq y) \rightarrow (\log(b,\ y) \not< \log(b,\ x))$

```
; these two lemmas are useful in time analysis.
```

THEOREM: ta-lemma-1
$(a \leq a1) \rightarrow ((x + (y * \log(2,\ a))) \leq (x + (y * \log(2,\ a1))))$

THEOREM: ta-lemma-2
$((a \leq a1) \wedge (b \leq b1))$
$\rightarrow \quad ((x + (y * (\log(2,\ a) + \log(2,\ b))))$
$\qquad \leq \quad (x + (y * (\log(2,\ a1) + \log(2,\ b1)))))$

EVENT: Make the library `"mc20-0"` and compile it.

Index