

#|

Copyright (C) 1994 by Yuan Yu. All Rights Reserved.

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

NO WARRANTY

Yuan Yu PROVIDES 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 Yuan Yu 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.

#|

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) ∈ y then subset(cdr(x), y)
         else fendif
    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) ∈ y then subbagp(cdr(x), delete(car(x), y))
         else fendif
    else t endif
```

; the difference.

DEFINITION:

```
bagdiff(x, y)
=  if listp(y)
    then if car(y) ∈ 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) ∈ 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

$(x \notin y) \rightarrow (\text{delete}(x, y) = y)$

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
 $(x < 1) = (x \simeq 0)$

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

; lemmas about plus.

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

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

$$((x + y) = x) = ((x \in \mathbf{N}) \wedge (y \simeq 0))$$

THEOREM: plus-equal-cancel

$$((x + y) = (x + z)) = (\text{fix}(y) = \text{fix}(z))$$

THEOREM: plus-lessp-cancel-0

$$(x < (x + y)) = (y \not\simeq 0)$$

THEOREM: plus-lessp-cancel-1

$$((x + y) < (x + z)) = (y < z)$$

THEOREM: plus-lessp-cancel-add1

$$((y + x) < (1 + y)) = (x \simeq 0)$$

THEOREM: plus-equal-0

$$((x + y) = 0) = ((x \simeq 0) \wedge (y \simeq 0))$$

; lemmas about difference.

THEOREM: sub1-of-1

$$((x - 1) = 0) = ((x \simeq 0) \vee (x = 1))$$

THEOREM: difference-sub1

$$(x - 1) = (x - 1)$$

THEOREM: difference-sub1-sub1

$$\begin{aligned} & ((x - (y - 1)) - 1) \\ &= \text{if } y \simeq 0 \text{ then } x - 1 \\ & \quad \text{elseif } x < y \text{ then } 0 \\ & \quad \text{else } x - y \text{ endif} \end{aligned}$$

THEOREM: difference-0

$$(x \leq y) \rightarrow ((x - y) = 0)$$

THEOREM: difference-x-x

$$(x - x) = 0$$

THEOREM: difference-plus-cancel0

$$(((x + y) - x) = \text{fix}(y)) \wedge (((y + x) - x) = \text{fix}(y))$$

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

$$(y \not\prec z) \rightarrow ((x - (y - z)) = ((x + z) - y))$$

THEOREM: difference-plus-cancel1

$$((x + y) - (x + z)) = (y - z)$$

THEOREM: difference-plus-cancel-add1

$$((y + x) - (1 + y)) = (x - 1)$$

THEOREM: difference-lessp

$$((m - n) < m) = ((m \not\prec 0) \wedge (n \not\prec 0))$$

THEOREM: difference-lessp1

$$(x < z) \rightarrow (((x - y) < z) = \mathbf{t})$$

THEOREM: difference=0

$$(0 = (x - y)) = (y \not\prec x)$$

THEOREM: difference-equal-cancel-0

$$(x = (x - y)) = ((x \in \mathbf{N}) \wedge ((x = 0) \vee (y \simeq 0)))$$

THEOREM: difference-equal-cancel-1

$$\begin{aligned} & ((x - z) = (y - z)) \\ = & \text{ if } x < z \text{ then } z \not\prec y \\ & \text{ elseif } y < z \text{ then } z \not\prec x \\ & \text{ else fix}(x) = \text{fix}(y) \text{ endif} \end{aligned}$$

THEOREM: difference-lessp-cancel

$$\begin{aligned} & ((a - c) < (b - c)) \\ = & \text{ if } c \leq a \text{ then } a < b \\ & \text{ else } c < b \text{ endif} \end{aligned}$$

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

DEFINITION:

plus-fringe(x)

$$\begin{aligned} = & \text{ if listp}(x) \wedge (\text{car}(x) = \text{'plus}) \\ & \text{ then append}(\text{plus-fringe}(\text{cadr}(x)), \text{plus-fringe}(\text{caddr}(x))) \\ & \text{ else cons}(x, \text{nil}) \text{ endif} \end{aligned}$$

DEFINITION:

plus-tree(l)

$$\begin{aligned} = & \text{ if } l \simeq \text{nil} \text{ then } ''0 \\ & \text{ elseif cdr}(l) \simeq \text{nil} \text{ then list}(\text{'fix}, \text{car}(l)) \\ & \text{ elseif caddr}(l) \simeq \text{nil} \text{ then list}(\text{'plus}, \text{car}(l), \text{cadr}(l)) \\ & \text{ else list}(\text{'plus}, \text{car}(l), \text{plus-tree}(\text{cdr}(l))) \text{ endif} \end{aligned}$$

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

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

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

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

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

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

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
 $(\text{car}(\text{eval\$}(\mathbf{t}, \text{'list}, x, a)) = \text{eval\$}(\mathbf{t}, \text{car}(x), a))$
 $\wedge (\text{cdr}(\text{eval\$}(\mathbf{t}, \text{'list}, x, a))$
 $= \text{ if listp}(x) \text{ then } \text{eval\$}(\mathbf{t}, \text{cdr}(x), a)$
 $\text{ else } 0 \text{ endif})$

THEOREM: eval\$-quote
 $\text{eval}(\mathbf{t}, \text{cons}(\text{'quote}, \text{args}), a) = \text{car}(\text{args})$

THEOREM: listp-eval\$
 $\text{listp}(\text{eval}(\text{'list}, x, a)) = \text{listp}(x)$

; the meta lemma to cancel identical plus terms in equality. For example,
 $(\text{EQUAL}(\text{PLUS A B C})(\text{PLUS B D E})) \Rightarrow (\text{EQUAL}(\text{PLUS A C})(\text{PLUS D E}))$.

DEFINITION:

cancel-equal-plus(x)
 $=$ **if** listp(x) $\wedge$ (car(x) = 'equal)
 then if listp(cadr(x))
 $\wedge$ (caadr(x) = 'plus)
 $\wedge$ listp(caddr(x))
 $\wedge$ (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))
 $\wedge$ (caadr(x) = 'plus)
 $\wedge$ (caddr(x) $\in$ 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))
 $\wedge$ (caaddr(x) = 'plus)
 $\wedge$ (cadr(x) $\in$ 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
 $\text{eval}(\mathbf{t}, x, a) = \text{eval}(\mathbf{t}, \text{cancel-equal-plus}(x), a)$

; 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)),
                                         plus-fringe (cadr (x))))))
        else x endif
```

```

                                bagint (plus-fringe (cadr (x)),
                                      plus-fringe (caddr (x))))))
elseif listp (cadr (x))
  ∧ (caadr (x) = 'plus)
  ∧ (caddr (x) ∈ plus-fringe (cadr (x)))
then list ('quote, f)
elseif listp (caddr (x))
  ∧ (caaddr (x) = 'plus)
  ∧ (cadr (x) ∈ plus-fringe (caddr (x)))
then list ('not,
          list ('zerop,
              plus-tree (delete (cadr (x), plus-fringe (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
 $((x \simeq 0) \vee (y \simeq 0)) \rightarrow ((x * y) = 0)$

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
 $((x * y) = 0) = ((x \simeq 0) \vee (y \simeq 0))$

THEOREM: times-equal-1
 $((x * y) = 1) = ((x = 1) \wedge (y = 1))$

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

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

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
 $((x * y) = y)$
 $= ((y \in \mathbf{N})$
 $\quad \wedge \text{ if } y = 0 \text{ then } t$
 $\quad \quad \text{ else } x = 1 \text{ endif}))$
 $\wedge ((y * x) = y)$
 $= ((y \in \mathbf{N})$
 $\quad \wedge \text{ if } y = 0 \text{ then } t$
 $\quad \quad \text{ else } x = 1 \text{ endif}))$

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** $x \simeq \text{nil}$ **then** ''1
 elseif cdr(x) $\simeq \text{nil}$ **then** list('fix, car(x))
 elseif caddr(x) $\simeq \text{nil}$ **then** list('times, car(x), cadr(x))
 else list('times, car(x), times-tree(cdr(x))) **endif**

DEFINITION:
and-not-zerop-tree(x)
= **if** $x \simeq \text{nil}$ **then** '(true)
 elseif cdr(x) $\simeq \text{nil}$ **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\$(t, x , a) $\in \mathbf{N}$)

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

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

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

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

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
 $(\text{eval\$}(\mathbf{t}, \text{and-not-zerop-tree}(x), a) \wedge \text{subbagp}(x, y) \wedge \text{subbagp}(x, z))$
 $\rightarrow ((\text{eval\$}(\mathbf{t}, \text{times-tree}(\text{bagdiff}(y, x)), a)$
 $< \text{eval\$}(\mathbf{t}, \text{times-tree}(\text{bagdiff}(z, x)), a))$
 $= (\text{eval\$}(\mathbf{t}, \text{times-tree}(y), a) < \text{eval\$}(\mathbf{t}, \text{times-tree}(z), a)))$

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

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

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

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

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

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

THEOREM: lessp-1-times-tree-delete

$(w \in x)$
 $\rightarrow ((1 < \text{eval\$}(\mathbf{t}, \text{times-tree}(\text{delete}(w, x)), a))$
 $= \text{if } \text{eval\$}(\mathbf{t}, w, a) \simeq 0 \text{ then lessp-1-times-tree-delete-end}(w, x, a)$
 $\text{else } \text{eval\$}(\mathbf{t}, w, a) < \text{eval\$}(\mathbf{t}, \text{times-tree}(x), a) \text{ endif}$

THEOREM: eval\$-times-fringe-member-zero

$((e \in \text{times-fringe}(\text{cons}(\mathbf{'times}, x))) \wedge (\text{eval\$}(\mathbf{t}, e, a) \simeq 0))$
 $\rightarrow ((\text{eval\$}(\mathbf{t}, \text{car}(x), a) * \text{eval\$}(\mathbf{t}, \text{cadr}(x), a)) = 0)$

DEFINITION:

cancel-lessp-times(x)
 $= \text{if listp}(x) \wedge (\text{car}(x) = \mathbf{'lessp})$
 $\text{then if } (\text{caadr}(x) = \mathbf{'times}) \wedge (\text{caaddr}(x) = \mathbf{'times})$
 $\text{then if listp}(\text{bagint}(\text{times-fringe}(\text{cadr}(x)),$
 $\text{times-fringe}(\text{caddr}(x))))$
 $\text{then list}(\mathbf{'and},$
 $\text{and-not-zerop-tree}(\text{bagint}(\text{times-fringe}(\text{cadr}(x)),$
 $\text{times-fringe}(\text{caddr}(x)))),$
 $\text{list}(\mathbf{'lessp},$
 $\text{times-tree}(\text{bagdiff}(\text{times-fringe}(\text{cadr}(x)),$
 $\text{bagint}(\text{times-fringe}(\text{cadr}(x)),$
 $\text{times-fringe}(\text{caddr}(x))))),$
 $\text{times-tree}(\text{bagdiff}(\text{times-fringe}(\text{caddr}(x)),$
 $\text{bagint}(\text{times-fringe}(\text{cadr}(x)),$
 $\text{times-fringe}(\text{caddr}(x))))))$
 $\text{else } x \text{ endif}$
 $\text{elseif listp}(\text{cadr}(x))$
 $\wedge (\text{caadr}(x) = \mathbf{'times})$
 $\wedge (\text{caddr}(x) \in \text{times-fringe}(\text{cadr}(x)))$
 $\text{then list}(\mathbf{'and},$
 $\text{list}(\mathbf{'not}, \text{list}(\mathbf{'zerop}, \text{caddr}(x))),$
 $\text{list}(\mathbf{'not},$
 $\text{and-not-zerop-tree}(\text{delete}(\text{caddr}(x),$
 $\text{times-fringe}(\text{cadr}(x))))))$
 $\text{elseif listp}(\text{caddr}(x))$
 $\wedge (\text{caaddr}(x) = \mathbf{'times})$
 $\wedge (\text{cadr}(x) \in \text{times-fringe}(\text{caddr}(x)))$
 $\text{then list}(\mathbf{'and},$
 $\text{list}(\mathbf{'not}, \text{list}(\mathbf{'zerop}, \text{cadr}(x))),$
 $\text{list}(\mathbf{'lessp},$
 $\mathbf{'1},$
 $\text{times-tree}(\text{delete}(\text{cadr}(x), \text{times-fringe}(\text{caddr}(x))))))$
 $\text{else } x \text{ endif}$
 $\text{else } x \text{ endif}$

```
; 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
 $\text{eval}\$ (\mathbf{t}, x, a) = \text{eval}\$ (\mathbf{t}, \text{cancel-lessp-times}(x), a)$

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

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

THEOREM: eval\$-equal-times-tree-bagdiff

$$\begin{aligned} & (\text{eval}\$ (\mathbf{t}, \text{and-not-zerop-tree}(x), a) \wedge \text{subbagp}(x, y) \wedge \text{subbagp}(x, z)) \\ \rightarrow & ((\text{eval}\$ (\mathbf{t}, \text{times-tree}(\text{bagdiff}(y, x)), a) \\ & = \text{eval}\$ (\mathbf{t}, \text{times-tree}(\text{bagdiff}(z, x)), a)) \\ & = (\text{eval}\$ (\mathbf{t}, \text{times-tree}(y), a) = \text{eval}\$ (\mathbf{t}, \text{times-tree}(z), a))) \end{aligned}$$

THEOREM: zerop-makes-equal-true-bridge

$$\begin{aligned} & (\neg \text{eval}\$ (\mathbf{t}, \\ & \quad \text{and-not-zerop-tree}(\text{bagint}(\text{times-fringe}(\text{cons}('times, x)), \\ & \quad \quad \text{times-fringe}(\text{cons}('times, y)))), \\ & \quad a)) \\ \rightarrow & (((\text{eval}\$ (\mathbf{t}, \text{car}(x), a) * \text{eval}\$ (\mathbf{t}, \text{cadr}(x), a)) \\ & = (\text{eval}\$ (\mathbf{t}, \text{car}(y), a) * \text{eval}\$ (\mathbf{t}, \text{cadr}(y), a))) \\ & = \mathbf{t}) \end{aligned}$$

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

THEOREM: equal-1-times-tree-delete

$$\begin{aligned} & (w \in x) \\ \rightarrow & ((\text{eval}\$ (\mathbf{t}, \text{times-tree}(\text{delete}(w, x)), a) = 1) \\ & = \text{if } \text{eval}\$ (\mathbf{t}, w, a) \simeq 0 \\ & \quad \text{then } \text{equal-1-eval}\$ \text{-times-tree-delete-end}(w, x, a) \\ & \quad \text{else } \text{eval}\$ (\mathbf{t}, \text{times-tree}(x), a) = \text{eval}\$ (\mathbf{t}, w, a) \text{ endif}) \end{aligned}$$

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

```

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
 $\text{eval}\$(\mathbf{t}, x, a) = \text{eval}\$(\mathbf{t}, \text{cancel-equal-times}(x), a)$

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

; lemmas about exp.

DEFINITION:

$\text{exp}(x, y)$
= **if** $y \simeq 0$ **then** 1
 else $x * \text{exp}(x, y - 1)$ **endif**

THEOREM: exp-of-0

$\text{exp}(0, k)$
= **if** $k \simeq 0$ **then** 1
 else 0 **endif**

THEOREM: exp-of-1

$\text{exp}(1, k) = 1$

THEOREM: exp-plus

$(\text{exp}(x, y) * \text{exp}(x, z)) = \text{exp}(x, y + z)$

THEOREM: exp-times

$\text{exp}(x * y, z) = (\text{exp}(x, z) * \text{exp}(y, z))$

THEOREM: exp-exp

$\text{exp}(\text{exp}(x, y), z) = \text{exp}(x, y * z)$

THEOREM: exp-of-2-0

$(m \not\simeq 0) \rightarrow (\text{exp}(m, n) \not\prec 1)$

THEOREM: exp-of-2-1

$(1 < \text{exp}(2, n)) = (n \not\simeq 0)$

THEOREM: exp-lessp

$(\text{exp}(x, y) < \text{exp}(x, z))$
= **if** $x \simeq 0$ **then** $(y \not\simeq 0) \wedge (z \simeq 0)$
 elseif $x = 1$ **then** f
 else $y < z$ **endif**

EVENT: Disable times.

THEOREM: times-exp2-lessp

$((i * \text{exp}(2, j)) < \text{exp}(2, k)) = (i < \text{exp}(2, k - j))$

; lemmas about remainder and quotient.

THEOREM: remainder-exit

$(i < j) \rightarrow ((i \bmod j) = \text{fix}(i))$

THEOREM: quotient-exit

$$(i < j) \rightarrow ((i \div j) = 0)$$

THEOREM: remainder-0

$$((0 \bmod x) = 0) \wedge ((x \bmod 0) = \text{fix}(x))$$

THEOREM: quotient-0

$$((0 \div x) = 0) \wedge ((x \div 0) = 0)$$

THEOREM: remainder-1

$$\begin{aligned} & ((1 \bmod x) \\ &= \text{if } x = 1 \text{ then } 0 \\ & \quad \text{else } 1 \text{ endif}) \\ & \wedge ((x \bmod 1) = 0) \end{aligned}$$

THEOREM: quotient-1

$$\begin{aligned} & ((1 \div x) \\ &= \text{if } x = 1 \text{ then } 1 \\ & \quad \text{else } 0 \text{ endif}) \\ & \wedge ((m \div 1) = \text{fix}(m)) \end{aligned}$$

THEOREM: remainder-x-x

$$(x \bmod x) = 0$$

THEOREM: quotient-x-x

$$\begin{aligned} & (x \div x) \\ &= \text{if } x \simeq 0 \text{ then } 0 \\ & \quad \text{else } 1 \text{ endif} \end{aligned}$$

THEOREM: quotient-equal-0

$$((m \div n) = 0) = ((m \simeq 0) \vee (n \simeq 0) \vee (m < n))$$

THEOREM: remainder-2x

$$((x + x) \bmod 2) = 0$$

THEOREM: quotient-2x

$$((x + x) \div 2) = \text{fix}(x)$$

THEOREM: remainder-2x-add1

$$((x + (1 + x)) \bmod 2) = 1$$

THEOREM: quotient-2x-add1

$$((x + (1 + x)) \div 2) = \text{fix}(x)$$

; A generalization lemma about quotient.

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

EVENT: Disable quotient-generalize.

THEOREM: remainder-lessp
 $((x \bmod y) < y) = (y \neq 0)$

THEOREM: quotient-lessp
 $((m \div n) < m) = ((m \neq 0) \wedge ((n \simeq 0) \vee (n \neq 1)))$

THEOREM: remainder-lessp-linear
 $(y \neq 0) \rightarrow ((x \bmod y) < y)$

THEOREM: quotient-lessp-linear
 $((x \neq 0) \wedge (1 < y)) \rightarrow ((x \div y) < x)$

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

THEOREM: remainder-wrt-2
 $(n \bmod 2) < 2$

THEOREM: remainder-plus1
 $((i \bmod j) = 0) \rightarrow (((x + i) \bmod j) = (x \bmod j))$

THEOREM: remainder-plus2
 $((i \bmod j) = 0) \rightarrow (((i + x) \bmod j) = (x \bmod j))$

THEOREM: remainder-times
 $((x * y) \bmod y) = 0 \wedge ((y * x) \bmod y) = 0$

THEOREM: remainder-plus-times1
 $((x + (y * z)) \bmod y) = (x \bmod y)$

THEOREM: remainder-plus-times2
 $((x + (z * y)) \bmod y) = (x \bmod y)$

THEOREM: remainder-plus-plus
 $((i \bmod j) = 0) \rightarrow (((x + y + i) \bmod j) = ((x + y) \bmod j))$

THEOREM: remainder-plus-add1
 $((i \bmod j) = 0) \rightarrow (((1 + (x + i)) \bmod j) = ((1 + x) \bmod j))$

THEOREM: remainder-plus-difference1

$$\begin{aligned} & ((x \not\prec z) \wedge ((y \bmod w) = 0)) \\ \rightarrow & (((x + y) - z) \bmod w) = ((x - z) \bmod w) \end{aligned}$$

THEOREM: remainder-plus-difference2

$$\begin{aligned} & ((y \not\prec z) \wedge ((x \bmod w) = 0)) \\ \rightarrow & (((x + y) - z) \bmod w) = ((y - z) \bmod w) \end{aligned}$$

THEOREM: remainder-plus-plus-times1

$$((x + w + (y * z)) \bmod y) = ((x + w) \bmod y)$$

THEOREM: remainder-plus-plus-times2

$$((x + w + (z * y)) \bmod y) = ((x + w) \bmod y)$$

THEOREM: remainder-difference

$$\begin{aligned} & ((y \bmod z) = 0) \\ \rightarrow & (((x - y) \bmod z) \\ & = \text{if } x < y \text{ then } 0 \\ & \text{else } x \bmod z \text{ endif}) \end{aligned}$$

THEOREM: remainder-difference-times1

$$\begin{aligned} & ((x - (y * z)) \bmod z) \\ = & \text{if } x < (y * z) \text{ then } 0 \\ & \text{else } x \bmod z \text{ endif} \end{aligned}$$

THEOREM: remainder-difference-times2

$$\begin{aligned} & ((x - (y * z)) \bmod y) \\ = & \text{if } x < (y * z) \text{ then } 0 \\ & \text{else } x \bmod y \text{ endif} \end{aligned}$$

THEOREM: quotient-plus1

$$((i \bmod j) = 0) \rightarrow (((x + i) \div j) = ((x \div j) + (i \div j)))$$

THEOREM: quotient-plus2

$$((i \bmod j) = 0) \rightarrow (((i + x) \div j) = ((i \div j) + (x \div j)))$$

THEOREM: quotient-times

$$\begin{aligned} & (((x * y) \div y) \\ = & \text{if } y \simeq 0 \text{ then } 0 \\ & \text{else fix}(x) \text{ endif}) \\ \wedge & (((y * x) \div y) \\ = & \text{if } y \simeq 0 \text{ then } 0 \\ & \text{else fix}(x) \text{ endif}) \end{aligned}$$

THEOREM: quotient-plus-times1

$$\begin{aligned} & ((x + (y * z)) \div y) \\ = & ((x \div y) \\ & + \text{if } y \simeq 0 \text{ then } 0 \\ & \text{else fix}(z) \text{ endif}) \end{aligned}$$

THEOREM: quotient-plus-times2

$$\begin{aligned} & ((x + (z * y)) \div y) \\ = & ((x \div y) \\ & + \text{if } y \simeq 0 \text{ then } 0 \\ & \text{else fix}(z) \text{ endif}) \end{aligned}$$

THEOREM: quotient-plus-plus

$$\begin{aligned} & ((i \bmod j) = 0) \\ \rightarrow & (((x + y + i) \div j) = (((x + y) \div j) + (i \div j))) \end{aligned}$$

THEOREM: quotient-plus-add1

$$\begin{aligned} & ((i \bmod j) = 0) \\ \rightarrow & (((1 + (x + i)) \div j) = (((1 + x) \div j) + (i \div j))) \end{aligned}$$

THEOREM: quotient-difference-plus1

$$\begin{aligned} & ((x \not\leq z) \wedge ((y \bmod w) = 0)) \\ \rightarrow & (((x + y) - z) \div w) = (((x - z) \div w) + (y \div w)) \end{aligned}$$

THEOREM: quotient-difference-plus2

$$\begin{aligned} & ((y \not\leq z) \wedge ((x \bmod w) = 0)) \\ \rightarrow & (((x + y) - z) \div w) = (((y - z) \div w) + (x \div w)) \end{aligned}$$

THEOREM: quotient-difference

$$\begin{aligned} & ((y \bmod z) = 0) \\ \rightarrow & (((x - y) \div z) \\ = & \text{if } x < y \text{ then } 0 \\ & \text{else } (x \div z) - (y \div z) \text{ endif}) \end{aligned}$$

THEOREM: quotient-difference-times1

$$\begin{aligned} & ((x - (y * z)) \div z) \\ = & \text{if } x < (y * z) \text{ then } 0 \\ & \text{else } (x \div z) - \text{fix}(y) \text{ endif} \end{aligned}$$

THEOREM: quotient-difference-times2

$$\begin{aligned} & ((x - (y * z)) \div y) \\ = & \text{if } x < (y * z) \text{ then } 0 \\ & \text{else } (x \div y) - \text{fix}(z) \text{ endif} \end{aligned}$$

EVENT: Disable remainder-difference.

EVENT: Disable quotient-difference.

THEOREM: remainder-sub1

$$\begin{aligned}
& ((m - 1) \bmod n) \\
& = \text{if } n \simeq 0 \text{ then } m - 1 \\
& \quad \text{elseif } (m \bmod n) = 0 \\
& \quad \text{then if } m \simeq 0 \text{ then } 0 \\
& \quad \quad \text{else } n - 1 \text{ endif} \\
& \quad \text{else } (m \bmod n) - 1 \text{ endif}
\end{aligned}$$

THEOREM: quotient-sub1

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

THEOREM: remainder-quotient

$$((y * (x \div y)) + (x \bmod y)) = \text{fix}(x)$$

THEOREM: remainder-quotient-elim

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

THEOREM: remainder-add1

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

THEOREM: quotient-add1

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

THEOREM: times-plus-lessp

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

THEOREM: quotient-shrink-fast

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

THEOREM: remainder-plus-remainder1

$$((x + (y \bmod z)) \bmod z) = ((x + y) \bmod z)$$

THEOREM: remainder-difference-remainder1

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

THEOREM: remainder-plus-remainder2

$$((x + y + (z \bmod k)) \bmod k) = ((x + y + z) \bmod k)$$

THEOREM: remainder-plus-remainder

$$(((x \bmod z) + (y \bmod z)) \bmod z) = ((x + y) \bmod z)$$

THEOREM: remainder-crock

$$(y < z) \rightarrow (((y * x) \bmod (x * z)) = (y * x))$$

THEOREM: times-distributes-remainder

$$((x * y) \bmod (x * z)) = (x * (y \bmod z))$$

THEOREM: quotient-crock

$$(y < z) \rightarrow (((y * x) \div (x * z)) = 0)$$

THEOREM: quotient-times-cancel

$$\begin{aligned} & ((x * y) \div (x * z)) \\ &= \text{if } x \simeq 0 \text{ then } 0 \\ & \quad \text{else } y \div z \text{ endif} \end{aligned}$$

EVENT: Disable remainder-crock.

EVENT: Disable quotient-crock.

THEOREM: remainder-distributes-times2-add1

$$((1 + (2 * y)) \bmod (2 * z)) = (1 + (2 * (y \bmod z)))$$

THEOREM: quotient-distributes-times2-add1

$$((1 + (2 * y)) \div (2 * z)) = (y \div z)$$

THEOREM: quotient-exp

$$\begin{aligned} & (1 < i) \\ \rightarrow & ((\exp(i, j) \div \exp(i, k)) \\ &= \text{if } j < k \text{ then } 0 \\ & \quad \text{else } \exp(i, j - k) \text{ endif}) \end{aligned}$$

THEOREM: remainder-exp

$$\begin{aligned} & (1 < i) \\ \rightarrow & ((\exp(i, j) \bmod \exp(i, k)) \\ &= \text{if } j < k \text{ then } \exp(i, j) \\ & \quad \text{else } 0 \text{ endif}) \end{aligned}$$

THEOREM: remainder-plus-cancel0

$$\begin{aligned} & (((i + j) \bmod n) = i) \\ &= \text{if } n \simeq 0 \text{ then } (i \in \mathbf{N}) \wedge (j \simeq 0) \\ & \quad \text{elseif } i < n \text{ then } (i \in \mathbf{N}) \wedge ((j \bmod n) = 0) \\ & \quad \text{else f endif} \end{aligned}$$

THEOREM: remainder-plus-cancel

$$(((i + j) \bmod n) = ((i + k) \bmod n)) = ((j \bmod n) = (k \bmod n))$$

THEOREM: quotient-times-lessp

$$\begin{aligned} & ((x \div z) < y) \\ = & \text{ if } z \simeq 0 \text{ then } y \not\simeq 0 \\ & \text{ else } x < (z * y) \text{ endif} \end{aligned}$$

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)

$$\begin{aligned} = & \text{ if } i \simeq 0 \text{ then } t \\ & \text{ else quot2-sub12-induct } (x \div 2, y \div 2, i - 1, j - 1) \text{ endif} \end{aligned}$$

THEOREM: lessp-plus-times-exp2

$$\begin{aligned} & (x < \exp(2, i)) \\ \rightarrow & (((x + (y * \exp(2, i))) < \exp(2, n)) \\ = & \text{ if } y \simeq 0 \text{ then } x < \exp(2, n) \\ & \text{ else } y < \exp(2, n - i) \text{ endif} \end{aligned}$$

THEOREM: lessp-plus-exp2

$$(x < \exp(2, i)) \rightarrow (((x + \exp(2, i)) < \exp(2, n)) = (i < n))$$

THEOREM: remainder-times-exp2-1

$$((x * \exp(2, i)) \bmod \exp(2, j)) = ((x \bmod \exp(2, j - i)) * \exp(2, i))$$

THEOREM: remainder-times-exp2-2

$$((\exp(2, i) * x) \bmod \exp(2, j)) = ((x \bmod \exp(2, j - i)) * \exp(2, i))$$

; special cases of remainder-times-exp2.

THEOREM: remainder-times-exp2-3

$$\begin{aligned} & (((x * \exp(2, i)) \bmod 2) \\ = & \text{ if } i \simeq 0 \text{ then } x \bmod 2 \\ & \text{ else } 0 \text{ endif} \\ \wedge & (((\exp(2, i) * x) \bmod 2) \\ = & \text{ if } i \simeq 0 \text{ then } x \bmod 2 \\ & \text{ else } 0 \text{ endif} \end{aligned}$$

THEOREM: remainder-times-exp2-4

$$\begin{aligned}
& (((x * \exp(2, i) * y) \bmod 2) \\
&= \text{if } i \simeq 0 \text{ then } (x * y) \bmod 2 \\
&\quad \text{else } 0 \text{ endif}) \\
\wedge \quad & (((x * y * \exp(2, i)) \bmod 2) \\
&= \text{if } i \simeq 0 \text{ then } (x * y) \bmod 2 \\
&\quad \text{else } 0 \text{ endif})
\end{aligned}$$

THEOREM: quotient-times-exp2-1

$$\begin{aligned}
& ((x * \exp(2, i)) \div \exp(2, j)) \\
&= \text{if } i < j \text{ then } x \div \exp(2, j - i) \\
&\quad \text{else } x * \exp(2, i - j) \text{ endif}
\end{aligned}$$

THEOREM: quotient-times-exp2-2

$$\begin{aligned}
& ((\exp(2, i) * x) \div \exp(2, j)) \\
&= \text{if } i < j \text{ then } x \div \exp(2, j - i) \\
&\quad \text{else } x * \exp(2, i - j) \text{ endif}
\end{aligned}$$

THEOREM: quotient-times-exp2-3

$$\begin{aligned}
& (((x * \exp(2, i)) \div 2) \\
&= \text{if } i \simeq 0 \text{ then } x \div 2 \\
&\quad \text{else } x * \exp(2, i - 1) \text{ endif}) \\
\wedge \quad & (((\exp(2, i) * x) \div 2) \\
&= \text{if } i \simeq 0 \text{ then } x \div 2 \\
&\quad \text{else } x * \exp(2, i - 1) \text{ endif})
\end{aligned}$$

THEOREM: quotient-times-exp2-4

$$\begin{aligned}
& (((x * \exp(2, i) * y) \div 2) \\
&= \text{if } i \simeq 0 \text{ then } (x * y) \div 2 \\
&\quad \text{else } x * y * \exp(2, i - 1) \text{ endif}) \\
\wedge \quad & (((x * y * \exp(2, i)) \div 2) \\
&= \text{if } i \simeq 0 \text{ then } (x * y) \div 2 \\
&\quad \text{else } x * y * \exp(2, i - 1) \text{ endif})
\end{aligned}$$

THEOREM: remainder-remainder-exp2

$$\begin{aligned}
& ((x \bmod \exp(2, i)) \bmod \exp(2, j)) \\
&= \text{if } i < j \text{ then } x \bmod \exp(2, i) \\
&\quad \text{else } x \bmod \exp(2, j) \text{ endif}
\end{aligned}$$

; a lemma for the event add-evenp.

THEOREM: remainder-remainder-2

$$\begin{aligned}
& ((x \bmod \exp(2, i)) \bmod 2) \\
&= \text{if } i \simeq 0 \text{ then } 0 \\
&\quad \text{else } x \bmod 2 \text{ endif}
\end{aligned}$$

; 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

$$\begin{aligned} & (x \leq z) \\ \rightarrow & ((x < (y * z)) \\ & = \text{if } (y \simeq 0) \vee (z \simeq 0) \text{ then f} \\ & \quad \text{elseif } y = 1 \text{ then } x < z \\ & \quad \text{else t endif}) \end{aligned}$$

DEFINITION:

$$\begin{aligned} & \log(b, x) \\ = & \text{if } (b \simeq 0) \vee (b = 1) \text{ then } 0 \\ & \quad \text{elseif } x < b \text{ then } 0 \\ & \quad \text{else } 1 + \log(b, x \div b) \text{ endif} \end{aligned}$$

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

$$(1 < b) \rightarrow (\log(b, \exp(b, n)) = \text{fix}(n))$$

THEOREM: log-times-exp

$$\begin{aligned} & ((1 < b) \wedge (x \not\simeq 0)) \\ \rightarrow & ((\log(b, x * \exp(b, n)) = (n + \log(b, x))) \\ & \quad \wedge (\log(b, \exp(b, n) * x) = (n + \log(b, x)))) \end{aligned}$$

THEOREM: log-times-exp-1

$$\begin{aligned} & ((1 < b) \wedge (x \not\simeq 0)) \\ \rightarrow & ((\log(b, x * b) = (1 + \log(b, x))) \\ & \quad \wedge (\log(b, b * x) = (1 + \log(b, x)))) \end{aligned}$$

THEOREM: log-quotient-exp

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

DEFINITION:

$$\begin{aligned} & \text{quotient2-induct}(b, x, y) \\ = & \text{if } (b \simeq 0) \vee (b = 1) \text{ then } 0 \\ & \quad \text{elseif } (x \simeq 0) \vee (y \simeq 0) \text{ then } 0 \\ & \quad \text{else quotient2-induct}(b, x \div b, y \div b) \text{ endif} \end{aligned}$$

THEOREM: log-leq

$$(x \leq y) \rightarrow (\log(b, y) \not\leq \log(b, x))$$

; 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

$$\begin{aligned} & ((a \leq a1) \wedge (b \leq b1)) \\ \rightarrow & ((x + (y * (\log(2, a) + \log(2, b)))) \\ & \leq (x + (y * (\log(2, a1) + \log(2, b1))))) \end{aligned}$$

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

Index

- and-not-zero-tree, 11–15
- and-not-zero-tree-delete, 12
- and-not-zero-tree-lessp, 12
- bagdiff, 2, 6–9, 12–15
- bagint, 2, 3, 7–9, 12–15
- boolean, 10
- bridge-to-subbagp-implies-plus-t
ree-geq, 6
- cadr-eval\$-list, 6
- cancel-difference-plus, 8
- cancel-equal-plus, 7
- cancel-equal-times, 14, 15
- cancel-lessp-plus, 8, 9
- cancel-lessp-times, 13, 14
- correctness-of-cancel-difference
-plus, 8
- correctness-of-cancel-equal-plu
s, 7
- correctness-of-cancel-equal-time
s, 15
- correctness-of-cancel-lessp-plu
s, 9
- correctness-of-cancel-lessp-time
s, 14
- delete, 2, 3, 6–9, 11–15
- delete-commutativity, 3
- delete-non-member, 2
- difference-0, 4
- difference-difference1, 5
- difference-difference2, 5
- difference-equal-cancel-0, 5
- difference-equal-cancel-1, 5
- difference-lessp, 5
- difference-lessp-cancel, 5
- difference-lessp1, 5
- difference-plus-cancel-add1, 5
- difference-plus-cancel0, 4
- difference-plus-cancel1, 5
- difference-plus1, 4
- difference-plus2, 4
- difference-sub1, 4
- difference-sub1-sub1, 4
- difference-x-x, 4
- difference=0, 5
- equal-1-eval\$-times-tree-delete
-end, 14
- equal-1-times-tree-delete, 14
- equal-iff, 10
- eval\$-and-not-zero-tree-end, 12
- eval\$-equal-times-tree-bagdiff, 14
- eval\$-lessp-times-tree-bagdiff, 12
- eval\$-plus-tree-append, 6
- eval\$-quote, 7
- eval\$-times-fringe-member-zero, 13
- eval\$-times-member, 11
- eval\$-times-tree-append, 12
- eval\$-times-tree-numberp, 11
- exp, 16, 22–25
- exp-exp, 16
- exp-lessp, 16
- exp-of-0, 16
- exp-of-1, 16
- exp-of-2-0, 16
- exp-of-2-1, 16
- exp-plus, 16
- exp-times, 16
- lessp-1-times-tree-delete, 13
- lessp-1-times-tree-delete-end, 12, 13
- lessp-of-1, 3
- lessp-plus-exp2, 23
- lessp-plus-times-exp2, 23
- lessp-sub1, 3
- listp-eval\$, 7
- log, 25, 26
- log-equal-0, 25
- log-exp, 25
- log-leq, 26

- log-of-0, 25
- log-of-1, 25
- log-quotient-exp, 25
- log-times-exp, 25
- log-times-exp-1, 25

- member-delete, 3
- member-implies-numberp, 6
- member-implies-plus-tree-greater-than-p, 6

- numberp-eval\$-bridge, 6
- numberp-eval\$-plus, 6
- numberp-eval\$-plus-tree, 6
- numberp-eval\$-times, 11

- plus-add1, 3
- plus-add1-1, 3
- plus-associativity, 3
- plus-commutativity, 3
- plus-commutativity1, 3
- plus-equal-0, 4
- plus-equal-cancel, 4
- plus-equal-cancel0, 4
- plus-fringe, 5–9
- plus-lessp-cancel-0, 4
- plus-lessp-cancel-1, 4
- plus-lessp-cancel-2, 9
- plus-lessp-cancel-add1, 4
- plus-tree, 5–9
- plus-tree-bagdiff, 6
- plus-tree-delete, 6
- plus-tree-plus-fringe, 6

- quot2-sub12-induct, 23
- quotient-0, 17
- quotient-1, 17
- quotient-2x, 17
- quotient-2x-add1, 17
- quotient-add1, 21
- quotient-crock, 22
- quotient-difference, 20
- quotient-difference-plus1, 20
- quotient-difference-plus2, 20
- quotient-difference-times1, 20
- quotient-difference-times2, 20
- quotient-distributes-times2-add1, 22
- quotient-equal-0, 17
- quotient-exit, 17
- quotient-exp, 22
- quotient-generalize, 18
- quotient-leq, 18
- quotient-lessp, 18
- quotient-lessp-linear, 18
- quotient-plus-add1, 20
- quotient-plus-plus, 20
- quotient-plus-times1, 20
- quotient-plus-times2, 20
- quotient-plus1, 19
- quotient-plus2, 19
- quotient-quotient, 23
- quotient-shrink-fast, 21
- quotient-sub1, 21
- quotient-times, 19
- quotient-times-cancel, 22
- quotient-times-exp2-1, 24
- quotient-times-exp2-2, 24
- quotient-times-exp2-3, 24
- quotient-times-exp2-4, 24
- quotient-times-lessp, 23
- quotient-x-x, 17
- quotient2-induct, 25

- remainder-0, 17
- remainder-1, 17
- remainder-2x, 17
- remainder-2x-add1, 17
- remainder-add1, 21
- remainder-crock, 22
- remainder-difference, 19
- remainder-difference-remainder1, 21
- remainder-difference-times1, 19
- remainder-difference-times2, 19
- remainder-distributes-times2-add1, 22
- remainder-exit, 16
- remainder-exp, 22

remainder-lessp, 18
 remainder-lessp-linear, 18
 remainder-plus-add1, 18
 remainder-plus-cancel, 23
 remainder-plus-cancel0, 22
 remainder-plus-difference1, 19
 remainder-plus-difference2, 19
 remainder-plus-plus, 18
 remainder-plus-plus-times1, 19
 remainder-plus-plus-times2, 19
 remainder-plus-remainder, 22
 remainder-plus-remainder1, 21
 remainder-plus-remainder2, 21
 remainder-plus-times1, 18
 remainder-plus-times2, 18
 remainder-plus1, 18
 remainder-plus2, 18
 remainder-quotient, 21
 remainder-quotient-elim, 21
 remainder-remainder-2, 24
 remainder-remainder-exp2, 24
 remainder-sub1, 21
 remainder-times, 18
 remainder-times-exp2-1, 23
 remainder-times-exp2-2, 23
 remainder-times-exp2-3, 23
 remainder-times-exp2-4, 24
 remainder-wrt-2, 18
 remainder-x-x, 17

 sub1-of-1, 4
 subbagp, 2, 3, 6, 12, 14
 subbagp-bagint1, 3
 subbagp-bagint2, 3
 subbagp-cdr1, 3
 subbagp-cdr2, 3
 subbagp-delete, 3
 subbagp-implies-plus-tree-geq, 6
 subset, 2

 ta-lemma-1, 26
 ta-lemma-2, 26
 times-1, 9
 times-add1, 9
 times-add1-sub1, 10
 times-associativity, 10
 times-commutativity, 9
 times-commutativity1, 9
 times-distributes-difference, 11
 times-distributes-difference1, 11
 times-distributes-plus, 9
 times-distributes-plus-new, 23
 times-distributes-remainder, 22
 times-equal-0, 9
 times-equal-1, 9
 times-equal-cancel, 10
 times-equal-cancel0, 10
 times-exp2-lessp, 16
 times-fringe, 11–15
 times-lessp, 25
 times-lessp-0, 10
 times-lessp-1, 10
 times-lessp-cancel, 10
 times-lessp-cancel-1, 10
 times-lessp-cancel0, 10
 times-lessp-linear, 11
 times-plus-lessp, 21
 times-tree, 11–15
 times-tree-times-fringe, 12
 times-zero, 9
 times2-add1-lessp-cancel, 11

 zerop-makes-equal-true-bridge, 14
 zerop-makes-lessp-false-bridge, 12
 zerop-makes-times-tree-zero, 12