```
; William R. Bevier, Matt Kaufmann, and Matt Wilding
```

EVENT: Start with the library "bags" using the compiled version.

```
;; Jan 91 - some additional facts mostly having to do with GCD
;; added by MMW

;; Oct 90 - modified by MMW to eliminate theories

;; Tue Sep 26 10:20:45 1989, from ~wilding/numerical/newnat.events

;; NATURALS Theory

;; Created by Bill Bevier 1988 (see CLI internal note 057)

;; Modifications by Bill Bevier and Matt Wilding (9/89) including
;; adding some new metalemmas for times, reorganizing the theories,
;; removing some extraneous lemmas, and removing dependence upon
;; other theories (by adding the pertinent lemmas).
```

```
;; This script requires the bags theory

;; This script sets up a theory for the NATURALS with the following subtheories
;; ADDITION
;; MULTIPLICATION
;; REMAINDER
;; QUOTIENT
;; EXPONENTIATION
;; LOGS
;; GCDS

;; The theories of EXPONENTIATION, LOGS, and GCDS still need a lot of work

; -------------------------------------------------------------------------------
; ARITHMETIC
; -------------------------------------------------------------------------------

; -------------------- PLUS & DIFFERENCE --------------------

; ---------- EQUAL ----------
```

THEOREM: equal-plus-0
$((a + b) = 0) = ((a \simeq 0) \wedge (b \simeq 0))$

THEOREM: plus-cancellation
$((a + b) = (a + c)) = (\text{fix}(b) = \text{fix}(c))$

EVENT: Disable plus-cancellation.

THEOREM: equal-difference-0
$(((x - y) = 0) = (y \not< x)) \wedge ((0 = (x - y)) = (y \not< x))$

THEOREM: difference-cancellation
$((x - y) = (z - y))$
$= \quad \textbf{if } x < y \textbf{ then } y \not< z$
$\quad\quad \textbf{elseif } z < y \textbf{ then } y \not< x$
$\quad\quad \textbf{else } \text{fix}(x) = \text{fix}(z) \textbf{ endif}$

EVENT: Disable difference-cancellation.

```
; ---------- PLUS ----------
```

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

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

THEOREM: plus-zero-arg2
$(y \simeq \mathtt{0}) \rightarrow ((x + y) = \mathrm{fix}\,(x))$

THEOREM: plus-add1-arg1
$((1 + a) + b) = (1 + (a + b))$

THEOREM: plus-add1-arg2
$(x + (1 + y))$
$= \;\; \textbf{if } y \in \mathbf{N} \textbf{ then } 1 + (x + y)$
$\quad\;\; \textbf{else } 1 + x \textbf{ endif}$

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

THEOREM: plus-difference-arg1
$((a - b) + c)$
$= \;\; \textbf{if } b < a \textbf{ then } (a + c) - b$
$\quad\;\; \textbf{else } \mathtt{0} + c \textbf{ endif}$

THEOREM: plus-difference-arg2
$(a + (b - c))$
$= \;\; \textbf{if } c < b \textbf{ then } (a + b) - c$
$\quad\;\; \textbf{else } a + \mathtt{0} \textbf{ endif}$

```
; ---------- DIFFERENCE-PLUS cancellation rules ----------
;
; Here are the basic canonicalization rules for differences of sums. These
; are subsumed by the meta lemmas and are therefore globally disabled.
; They are here merely to prove the meta lemmas.
```

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

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

EVENT: Disable difference-plus-cancellation.

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

THEOREM: difference-plus-plus-cancellation
$(((x + y) - (x + z)) = (y - z))$
$\land \quad (((y + x) - (x + z)) = (y - z))$
$\land \quad (((x + y) - (z + x)) = (y - z))$
$\land \quad (((y + x) - (z + x)) = (y - z))$

EVENT: Disable difference-plus-plus-cancellation.

THEOREM: difference-plus-plus-cancellation-hack
$((w + x + a) - (y + z + a)) = ((w + x) - (y + z))$

EVENT: Disable difference-plus-plus-cancellation-hack.

```
; Here are a few more facts about difference needed to prove the meta lemmas.
; These are disabled here. We re-prove them after the proof of the meta
; lemmas so that they will fire before the meta lemmas in subsequent proofs.
```

THEOREM: diff-sub1-arg2
$(a - (b - 1))$
$= \quad$ **if** $b \simeq 0$ **then** $\text{fix}(a)$
elseif $a < b$ **then** 0
else $1 + (a - b)$ **endif**

EVENT: Disable diff-sub1-arg2.

THEOREM: diff-diff-arg1
$((x - y) - z) = (x - (y + z))$

THEOREM: diff-diff-arg2
$(a - (b - c))$
$= \quad$ **if** $b < c$ **then** $\text{fix}(a)$
else $(a + c) - b$ **endif**

```
; diff-diff-diff should be removed, but since the hack lemmas for
; correctness-of-cancel-difference-plus are designed for it, we'll
; keep it around.
```

THEOREM: diff-diff-diff
$((b \le a) \land (d \le c))$
$\to \quad (((a - b) - (c - d)) = ((a + d) - (b + c)))$

EVENT: Disable diff-diff-diff.

THEOREM: difference-lessp-arg1
$(a < b) \rightarrow ((a - b) = \texttt{0})$

EVENT: Disable difference-lessp-arg1.

```
; ----------------------------------------------------------------------------------------
; Meta Lemmas to Cancel PLUS and DIFFERENCE expressions
; ----------------------------------------------------------------------------------------

; ---------- PLUS-TREE and PLUS-FRINGE ----------
```

DEFINITION:
plus-fringe (x)
$=$ **if** listp $(x) \wedge (\text{car}\,(x) = \texttt{'plus})$
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
(listp $(x) \wedge (\text{car}\,(x) = \texttt{'plus})) \rightarrow (\text{eval\$}\,(\mathbf{t},\ x,\ a) \in \mathbf{N})$

EVENT: Disable numberp-eval\$-plus.

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

EVENT: Disable numberp-eval\$-plus-tree.

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

EVENT: Disable member-implies-plus-tree-greatereqp.

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

Event: Disable plus-tree-delete.

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

Event: Disable subbagp-implies-plus-tree-greatereqp.

Theorem: plus-tree-bagdiff
$$\begin{array}{l} \text{subbagp}(x,\ y) \\ \rightarrow \quad (\text{eval\$}(\mathbf{t},\ \text{plus-tree}(\text{bagdiff}(y,\ x)),\ a) \\ \qquad = \quad (\text{eval\$}(\mathbf{t},\ \text{plus-tree}(y),\ a) - \text{eval\$}(\mathbf{t},\ \text{plus-tree}(x),\ a))) \end{array}$$

Event: Disable plus-tree-bagdiff.

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

Event: Disable numberp-eval\$-bridge.

Theorem: bridge-to-subbagp-implies-plus-tree-greatereqp
$$\begin{array}{l} (\text{subbagp}(y,\ \text{plus-fringe}(z)) \\ \ \wedge \quad (\text{eval\$}(\mathbf{t},\ z,\ a) = \text{eval\$}(\mathbf{t},\ \text{plus-tree}(\text{plus-fringe}(z)),\ a))) \\ \rightarrow \quad ((\text{eval\$}(\mathbf{t},\ z,\ a) < \text{eval\$}(\mathbf{t},\ \text{plus-tree}(y),\ a)) = \mathbf{f}) \end{array}$$

Event: Disable bridge-to-subbagp-implies-plus-tree-greatereqp.

Theorem: eval\$-plus-tree-append
$$\begin{array}{l} \text{eval\$}(\mathbf{t},\ \text{plus-tree}(\text{append}(x,\ y)),\ a) \\ = \quad (\text{eval\$}(\mathbf{t},\ \text{plus-tree}(x),\ a) + \text{eval\$}(\mathbf{t},\ \text{plus-tree}(y),\ a)) \end{array}$$

Event: Disable eval\$-plus-tree-append.

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

Event: Disable plus-tree-plus-fringe.

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

Event: Disable member-implies-numberp.

THEOREM: cadr-eval$-list

$(\text{car}(\text{eval\$}(\texttt{'list}, x, a)) = \text{eval\$}(\texttt{t}, \text{car}(x), a))$
$\land \quad (\text{cdr}(\text{eval\$}(\texttt{'list}, x, a))$
$\quad = \quad \textbf{if } \text{listp}(x) \textbf{ then } \text{eval\$}(\texttt{'list}, \text{cdr}(x), a)$
$\quad\quad \textbf{else } \texttt{0} \textbf{ endif})$

EVENT: Disable cadr-eval$-list.

THEOREM: eval$-quote

$\text{eval\$}(\texttt{t}, \text{cons}(\texttt{'quote}, \mathit{args}), a) = \text{car}(\mathit{args})$

EVENT: Disable eval$-quote.

THEOREM: listp-eval$

$\text{listp}(\text{eval\$}(\texttt{'list}, x, a)) = \text{listp}(x)$

EVENT: Disable listp-eval$.

```
; ---------- CANCEL PLUS ----------

; CANCEL-EQUAL-PLUS cancels identical terms in a term which is the equality
; of two sums. For example,
;
;   (EQUAL (PLUS A B C) (PLUS B D E)) => (EQUAL (PLUS A C) (PLUS D E))
;
```

DEFINITION:

cancel-equal-plus (x)

$= \quad \textbf{if } \text{listp}(x) \land (\text{car}(x) = \texttt{'equal})$
$\quad \textbf{then if } \text{listp}(\text{cadr}(x))$
$\quad\quad\quad \land \quad (\text{caadr}(x) = \texttt{'plus})$
$\quad\quad\quad \land \quad \text{listp}(\text{caddr}(x))$
$\quad\quad\quad \land \quad (\text{caaddr}(x) = \texttt{'plus})$
$\quad\quad \textbf{then } \text{list}(\texttt{'equal},$
$\quad\quad\quad \text{plus-tree}(\text{bagdiff}(\text{plus-fringe}(\text{cadr}(x)),$
$\quad\quad\quad\quad \text{bagint}(\text{plus-fringe}(\text{cadr}(x)),$
$\quad\quad\quad\quad\quad \text{plus-fringe}(\text{caddr}(x))))),$
$\quad\quad\quad \text{plus-tree}(\text{bagdiff}(\text{plus-fringe}(\text{caddr}(x)),$
$\quad\quad\quad\quad \text{bagint}(\text{plus-fringe}(\text{cadr}(x)),$
$\quad\quad\quad\quad\quad \text{plus-fringe}(\text{caddr}(x))))))$
$\quad\quad \textbf{elseif } \text{listp}(\text{cadr}(x))$
$\quad\quad\quad \land \quad (\text{caadr}(x) = \texttt{'plus})$

$\wedge$ (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
eval\$ (**t**, x, a) = eval\$ (**t**, cancel-equal-plus (x), a)

```
; ---------- CANCEL-DIFFERENCE-PLUS ----------

; CANCEL-DIFFERENCE-PLUS cancels identical terms in a term which is the
; difference of two sums. For example,
;
;  (DIFFERENCE (PLUS A B C) (PLUS B D E)) => (DIFFERENCE (PLUS A C) (PLUS D E))
;
; Using rewrite rules, we canonicalize terms involving  PLUS and DIFFERENCE
; to be the DIFFERENCE of two sums. Then CANCEL-DIFFERENCE-PLUS cancels out
; like terms.
```

DEFINITION:
cancel-difference-plus (x)
= **if** listp (x) $\wedge$ (car (x) = `'difference`)
then if listp (cadr (x))
$\wedge$ (caadr (x) = `'plus`)
$\wedge$ listp (caddr (x))
$\wedge$ (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))
$\wedge$ (caadr (x) = `'plus`)
$\wedge$ (caddr (x) $\in$ plus-fringe (cadr (x)))
then plus-tree (delete (caddr (x), plus-fringe (cadr (x))))
elseif listp (caddr (x))
$\wedge$ (caaddr (x) = `'plus`)
$\wedge$ (cadr (x) $\in$ 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)

```
; ---------- DIFFERENCE ----------

; Here are the rules for difference terms which we want to try before
; the meta lemmas. They help canonicalize terms to differences of sums.
```

THEOREM: difference-elim
$((y \in \mathbf{N}) \wedge (y \not< x)) \rightarrow ((x + (y - x)) = y)$

THEOREM: difference-leq-arg1
$(a \leq b) \rightarrow ((a - b) = 0)$

THEOREM: difference-add1-arg2
$(a - (1 + b))$
= **if** $b < a$ **then** $(a - b) - 1$
else 0 **endif**

THEOREM: difference-sub1-arg2
$(a - (b - 1))$
= **if** $b \simeq 0$ **then** fix (a)
elseif $a < b$ **then** 0
else $1 + (a - b)$ **endif**

THEOREM: difference-difference-arg1
$((x - y) - z) = (x - (y + z))$

THEOREM: difference-difference-arg2
$(a - (b - c))$
= **if** $b < c$ **then** fix (a)
else $(a + c) - b$ **endif**

THEOREM: difference-x-x
$(x - x) = 0$

```
; ---------- LESSP ----------
```

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

EVENT: Disable lessp-difference-cancellation.

```
; CANCEL-LESSP-PLUS cancels LESSP terms whose arguments are sums.
; Examples:
; (LESSP (PLUS A B C) (PLUS A C D)) -> (LESSP (FIX B) (FIX D))
; (LESSP A (PLUS A B))  -> (NOT (ZEROP (FIX B)))
; (LESSP (PLUS A B) A)  -> F
```

DEFINITION:
cancel-lessp-plus (x)
$=$ **if** listp (x) $\wedge$ (car (x) = `'lessp`)
then if listp (cadr (x))
$\wedge$ (caadr (x) = `'plus`)
$\wedge$ listp (caddr (x))
$\wedge$ (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)),
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 (`'quote`, **f**)
elseif listp (caddr (x))
$\wedge$ (caaddr (x) = `'plus`)
$\wedge$ (cadr (x) $\in$ 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
$\text{eval\$}(\mathbf{t}, x, a) = \text{eval\$}(\mathbf{t}, \text{cancel-lessp-plus}(x), a)$

```
; ---------- TIMES ----------
```

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

THEOREM: equal-times-1
$((a * b) = 1) = ((a = 1) \wedge (b = 1))$

```
;(prove-lemma equal-sub1-times-0 (rewrite)
;       (equal (equal (sub1 (times a b)) 0)
;              (or (zerop a)
;                  (zerop b)
;                  (and (equal a 1) (equal b 1)))))
```

THEOREM: equal-sub1-0
$((x - 1) = 0) = ((x \simeq 0) \vee (x = 1))$

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

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

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

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

THEOREM: times-distributes-over-plus
$((x * (y + z)) = ((x * y) + (x * z)))$
$\wedge \quad (((x + y) * z) = ((x * z) + (y * z)))$

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

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

THEOREM: times-distributes-over-difference-proof
$((a - b) * c) = ((a * c) - (b * c))$

THEOREM: times-distributes-over-difference
$(((a - b) * c) = ((a * c) - (b * c)))$
$\land \quad ((a * (b - c)) = ((a * b) - (a * c)))$

THEOREM: times-quotient-proof
$((x \not\simeq \mathsf{0}) \land ((y \ \mathbf{mod}\ x) = \mathsf{0})) \rightarrow (((y \div x) * x) = \mathrm{fix}\,(y))$

THEOREM: times-quotient
$((y \not\simeq \mathsf{0}) \land ((x \ \mathbf{mod}\ y) = \mathsf{0}))$
$\rightarrow \quad ((((x \div y) * y) = \mathrm{fix}\,(x)) \land ((y * (x \div y)) = \mathrm{fix}\,(x)))$

THEOREM: times-1-arg1
$(\mathsf{1} * x) = \mathrm{fix}\,(x)$

THEOREM: lessp-times1-proof
$((a < b) \land (c \not\simeq \mathsf{0})) \rightarrow ((a < (b * c)) = \mathbf{t})$

THEOREM: lessp-times1
$((a < b) \land (c \not\simeq \mathsf{0}))$
$\rightarrow \quad (((a < (b * c)) = \mathbf{t}) \land ((a < (c * b)) = \mathbf{t}))$

THEOREM: lessp-times2-proof
$((a \leq b) \land (c \not\simeq \mathsf{0})) \rightarrow (((b * c) < a) = \mathbf{f})$

THEOREM: lessp-times2
$((a \leq b) \land (c \not\simeq \mathsf{0}))$
$\rightarrow \quad ((((b * c) < a) = \mathbf{f}) \land (((c * b) < a) = \mathbf{f}))$

THEOREM: lessp-times3-proof1
$((a \not\simeq \mathsf{0}) \land (\mathsf{1} < b)) \rightarrow (a < (a * b))$

THEOREM: lessp-times3-proof2
$(a < (a * b)) \rightarrow ((a \not\simeq \mathsf{0}) \land (\mathsf{1} < b))$

THEOREM: lessp-times3
$((a < (a * b)) = ((a \not\simeq \mathsf{0}) \land (\mathsf{1} < b)))$
$\land \quad ((a < (b * a)) = ((a \not\simeq \mathsf{0}) \land (\mathsf{1} < b)))$

THEOREM: lessp-times-cancellation-proof
$((x * z) < (y * z)) = ((z \not\simeq \mathsf{0}) \land (x < y))$

THEOREM: lessp-times-cancellation1

$$(((x * z) < (y * z)) = ((z \not\simeq 0) \wedge (x < y)))$$
$$\wedge \quad (((z * x) < (y * z)) = ((z \not\simeq 0) \wedge (x < y)))$$
$$\wedge \quad (((x * z) < (z * y)) = ((z \not\simeq 0) \wedge (x < y)))$$
$$\wedge \quad (((z * x) < (z * y)) = ((z \not\simeq 0) \wedge (x < y)))$$

EVENT: Disable lessp-times-cancellation1.

THEOREM: lessp-plus-times-proof

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

THEOREM: lessp-plus-times1

$$(((a + (b * c)) < b) = ((a < b) \wedge (c \simeq 0)))$$
$$\wedge \quad (((a + (c * b)) < b) = ((a < b) \wedge (c \simeq 0)))$$
$$\wedge \quad ((((c * b) + a) < b) = ((a < b) \wedge (c \simeq 0)))$$
$$\wedge \quad ((((b * c) + a) < b) = ((a < b) \wedge (c \simeq 0)))$$

THEOREM: lessp-plus-times2

$$((a \not\simeq 0) \wedge (x < a))$$
$$\rightarrow \quad ((((x + (a * b)) < (a * c)) = (b < c))$$
$$\wedge \quad (((x + (b * a)) < (a * c)) = (b < c))$$
$$\wedge \quad (((x + (a * b)) < (c * a)) = (b < c))$$
$$\wedge \quad (((x + (b * a)) < (c * a)) = (b < c))$$
$$\wedge \quad ((((a * b) + x) < (a * c)) = (b < c))$$
$$\wedge \quad ((((b * a) + x) < (a * c)) = (b < c))$$
$$\wedge \quad ((((a * b) + x) < (c * a)) = (b < c))$$
$$\wedge \quad ((((b * a) + x) < (c * a)) = (b < c)))$$

THEOREM: lessp-1-times

$$(1 < (a * b))$$
$$= \quad (\neg ((a \simeq 0) \vee (b \simeq 0) \vee ((a = 1) \wedge (b = 1))))$$

```
;;; meta lemmas to cancel lessp-times and equal-times expressions
;; examples
;; (lessp (times b (times d a)) (times b (times e (times a f)))) ->
;;                                             (and (and (not (zerop a))
;;                                                       (not (zerop b)))
;;                                                  (lessp (fix d) (times e f)))
;;
;; (equal (times b (times c d)) (times b d))    ->
;;                                            (or (or (zerop b) (zerop d))
;;                                                (equal (fix c) 1))
```

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:
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:
or-zerop-tree (x)
$=$ **if** $x \simeq$ **nil** **then** `'(false)`
elseif cdr $(x) \simeq$ **nil** **then** list (`'zerop`, car (x))
elseif cddr $(x) \simeq$ **nil**
then list (`'or`, list (`'zerop`, car (x)), list (`'zerop`, cadr (x)))
else list (`'or`, list (`'zerop`, car (x)), or-zerop-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) $\in \mathbf{N}$)

EVENT: Disable numberp-eval\$-times.

THEOREM: eval\$-times
(car (x) = `'times`)
$\rightarrow$ (eval\$ ($\mathbf{t}$, x, a) = (eval\$ ($\mathbf{t}$, cadr (x), a) $*$ eval\$ ($\mathbf{t}$, caddr (x), a)))

EVENT: Disable eval\$-times.

THEOREM: eval\$-or
(car (x) = `'or`)
$\rightarrow$ (eval\$ ($\mathbf{t}$, x, a) = (eval\$ ($\mathbf{t}$, cadr (x), a) $\vee$ eval\$ ($\mathbf{t}$, caddr (x), a)))

EVENT: Disable eval\$-or.

THEOREM: eval\$-equal
$(\text{car}(x) = \texttt{'equal})$
$\rightarrow \quad (\text{eval\$}(\mathbf{t}, x, a) = (\text{eval\$}(\mathbf{t}, \text{cadr}(x), a) = \text{eval\$}(\mathbf{t}, \text{caddr}(x), a)))$

EVENT: Disable eval\$-equal.

THEOREM: eval\$-lessp
$(\text{car}(x) = \texttt{'lessp})$
$\rightarrow \quad (\text{eval\$}(\mathbf{t}, x, a) = (\text{eval\$}(\mathbf{t}, \text{cadr}(x), a) < \text{eval\$}(\mathbf{t}, \text{caddr}(x), a)))$

EVENT: Disable eval\$-lessp.

THEOREM: eval\$-quotient
$(\text{car}(x) = \texttt{'quotient})$
$\rightarrow \quad (\text{eval\$}(\mathbf{t}, x, a) = (\text{eval\$}(\mathbf{t}, \text{cadr}(x), a) \div \text{eval\$}(\mathbf{t}, \text{caddr}(x), a)))$

EVENT: Disable eval\$-quotient.

THEOREM: eval\$-if
$(\text{car}(x) = \texttt{'if})$
$\rightarrow \quad (\text{eval\$}(\mathbf{t}, x, a)$
$\quad = \quad \textbf{if } \text{eval\$}(\mathbf{t}, \text{cadr}(x), a) \ \textbf{then } \text{eval\$}(\mathbf{t}, \text{caddr}(x), a)$
$\qquad \textbf{else } \text{eval\$}(\mathbf{t}, \text{cadddr}(x), a) \ \textbf{endif})$

EVENT: Disable eval\$-if.

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

EVENT: Disable numberp-eval\$-times-tree.

THEOREM: lessp-times-arg1
$(a \not\simeq \mathtt{0}) \rightarrow (((a * x) \not< (a * y)) = (x \not< y))$

THEOREM: infer-equality-from-not-lessp
$((a \in \mathbf{N}) \land (b \in \mathbf{N})) \rightarrow (((a \not< b) \land (b \not< a)) = (a = b))$

THEOREM: equal-times-arg1
$(a \not\simeq \mathtt{0}) \rightarrow (((a * x) = (a * y)) = (\text{fix}(x) = \text{fix}(y)))$

EVENT: Disable equal-times-arg1.

THEOREM: equal-times-bridge
$((a * b) = (c * (a * d))) = ((a \simeq \mathtt{0}) \lor (\text{fix}(b) = (c * d)))$

EVENT: Disable equal-times-bridge.

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)))

EVENT: Disable eval\$-times-member.

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

EVENT: Disable zerop-makes-times-tree-zero.

THEOREM: or-zerop-tree-is-not-zerop-tree
eval\$ (**t**, or-zerop-tree (x), a) = ($\neg$ eval\$ (**t**, and-not-zerop-tree (x), a))

EVENT: Disable or-zerop-tree-is-not-zerop-tree.

THEOREM: zerop-makes-times-tree-zero2
(eval\$ (**t**, or-zerop-tree (x), a) $\wedge$ subbagp (x, y))
$\rightarrow$ (eval\$ (**t**, times-tree (y), a) = `0`)

EVENT: Disable zerop-makes-times-tree-zero2.

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

EVENT: Disable times-tree-append.

THEOREM: times-tree-of-times-fringe
eval\$ (**t**, times-tree (times-fringe (x)), a) = fix (eval\$ (**t**, x, a))

EVENT: Disable times-tree-of-times-fringe.

DEFINITION:
cancel-lessp-times (x)
= **if** (car (x) = `'lessp`)
$\wedge$ (caadr (x) = `'times`)
$\wedge$ (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**
else x **endif**

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

EVENT: Disable eval\$-lessp-times-tree-bagdiff.

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

EVENT: Disable zerop-makes-lessp-false-bridge.

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

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

or-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)))))))
else x **endif**
else x **endif**

THEOREM: zerop-makes-equal-true-bridge
((car (x) = `'times`)
∧ (car (y) = `'times`)
∧ eval\$ (**t**, or-zerop-tree (bagint (times-fringe (x), times-fringe (y))), a))
→ (((eval\$ (**t**, cadr (x), a) $*$ eval\$ (**t**, caddr (x), a))
= (eval\$ (**t**, cadr (y), a) $*$ eval\$ (**t**, caddr (y), a)))
= **t**)

EVENT: Disable zerop-makes-equal-true-bridge.

THEOREM: eval\$-equal-times-tree-bagdiff
(subbagp (x, y) ∧ subbagp (x, z) ∧ (¬ eval\$ (**t**, or-zerop-tree (x), a)))
→ ((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)))

EVENT: Disable eval\$-equal-times-tree-bagdiff.

THEOREM: cancel-equal-times-preserves-inequality
(subbagp (z, x)
∧ subbagp (z, y)
∧ (eval\$ (**t**, times-tree (x), a) ≠ eval\$ (**t**, times-tree (y), a)))
→ (eval\$ (**t**, times-tree (bagdiff (x, z)), a)
≠ eval\$ (**t**, times-tree (bagdiff (y, z)), a))

EVENT: Disable cancel-equal-times-preserves-inequality.

THEOREM: cancel-equal-times-preserves-inequality-bridge
((car (x) = `'times`)
∧ (car (y) = `'times`)
∧ ((eval\$ (**t**, cadr (x), a) $*$ eval\$ (**t**, caddr (x), a))

$$\neq \quad (\text{eval\$}\,(\mathbf{t},\ \text{cadr}\,(y),\ a) * \text{eval\$}\,(\mathbf{t},\ \text{caddr}\,(y),\ a))))$$

$$\rightarrow \quad (\text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(\text{bagdiff}\,(\text{times-fringe}\,(x),\ \text{bagint}\,(\text{times-fringe}\,(x),\ \text{times-fringe}\,(y)))),\ a)$$

$$\neq \quad \text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(\text{bagdiff}\,(\text{times-fringe}\,(y),\ \text{bagint}\,(\text{times-fringe}\,(x),\ \text{times-fringe}\,(y)))),\ a))$$

EVENT: Disable cancel-equal-times-preserves-inequality-bridge.

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

```
; ---------- REMAINDER ----------
```

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

THEOREM: remainder-noop
$(a < b) \rightarrow ((a \ \mathbf{mod}\ b) = \text{fix}\,(a))$

THEOREM: remainder-of-non-number
$(a \notin \mathbf{N}) \rightarrow ((a \ \mathbf{mod}\ n) = (0 \ \mathbf{mod}\ n))$

THEOREM: remainder-zero
$(x \simeq 0) \rightarrow ((y \ \mathbf{mod}\ x) = \text{fix}\,(y))$

THEOREM: plus-remainder-times-quotient
$((x \ \mathbf{mod}\ y) + (y * (x \div y))) = \text{fix}\,(x)$

EVENT: Disable plus-remainder-times-quotient.

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

```
; (prove-lemma remainder-sub1 (rewrite)
;        (implies (and (not (zerop a))
;                      (not (zerop b)))
;                 (equal (remainder (sub1 a) b)
;                        (if (equal (remainder a b) 0)
;                            (sub1 b)
```

```
;                             (sub1 (remainder a b)))))
;         ((enable lessp-remainder
;                 remainder-noop
;                 remainder-quotient-elim)
;        (enable-theory addition)
;        (induct (remainder a b))))
```

THEOREM: remainder-add1
$((a \textbf{ mod } b) = 0) \rightarrow (((1 + a) \textbf{ mod } b) = (1 \textbf{ mod } b))$

THEOREM: remainder-plus-proof
$((b \textbf{ mod } c) = 0) \rightarrow (((a + b) \textbf{ mod } c) = (a \textbf{ mod } c))$

THEOREM: remainder-plus
$((a \textbf{ mod } c) = 0)$
$\rightarrow \quad ((((a + b) \textbf{ mod } c) = (b \textbf{ mod } c))$
$\qquad \wedge \quad (((b + a) \textbf{ mod } c) = (b \textbf{ mod } c))$
$\qquad \wedge \quad (((x + y + a) \textbf{ mod } c) = ((x + y) \textbf{ mod } c)))$

THEOREM: equal-remainder-plus-0-proof
$((a \textbf{ mod } c) = 0) \rightarrow ((((a + b) \textbf{ mod } c) = 0) = ((b \textbf{ mod } c) = 0))$

THEOREM: equal-remainder-plus-0
$((a \textbf{ mod } c) = 0)$
$\rightarrow \quad (((((a + b) \textbf{ mod } c) = 0) = ((b \textbf{ mod } c) = 0))$
$\qquad \wedge \quad ((((b + a) \textbf{ mod } c) = 0) = ((b \textbf{ mod } c) = 0))$
$\qquad \wedge \quad ((((x + y + a) \textbf{ mod } c) = 0)$
$\qquad\qquad = \quad (((x + y) \textbf{ mod } c) = 0)))$

THEOREM: equal-remainder-plus-remainder-proof
$(a < c) \rightarrow ((((a + b) \textbf{ mod } c) = (b \textbf{ mod } c)) = (a \simeq 0))$

THEOREM: equal-remainder-plus-remainder
$(a < c)$
$\rightarrow \quad (((((a + b) \textbf{ mod } c) = (b \textbf{ mod } c)) = (a \simeq 0))$
$\qquad \wedge \quad ((((b + a) \textbf{ mod } c) = (b \textbf{ mod } c)) = (a \simeq 0)))$

EVENT: Disable equal-remainder-plus-remainder.

THEOREM: remainder-times1-proof
$((b \textbf{ mod } c) = 0) \rightarrow (((a * b) \textbf{ mod } c) = 0)$

THEOREM: remainder-times1
$((b \textbf{ mod } c) = 0)$
$\rightarrow \quad ((((a * b) \textbf{ mod } c) = 0) \wedge (((b * a) \textbf{ mod } c) = 0))$

THEOREM: remainder-times1-instance-proof
$((x * y) \ \mathbf{mod}\ y) = 0$

THEOREM: remainder-times1-instance
$(((x * y) \ \mathbf{mod}\ y) = 0) \land (((x * y) \ \mathbf{mod}\ x) = 0)$

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

THEOREM: remainder-times-times
$(((x * y) \ \mathbf{mod}\ (x * z)) = (x * (y \ \mathbf{mod}\ z)))$
$\land \quad (((x * z) \ \mathbf{mod}\ (y * z)) = ((x \ \mathbf{mod}\ y) * z))$

EVENT: Disable remainder-times-times.

THEOREM: remainder-times2-proof
$((a \ \mathbf{mod}\ z) = 0) \rightarrow ((a \ \mathbf{mod}\ (z * y)) = (z * ((a \div z) \ \mathbf{mod}\ y)))$

THEOREM: remainder-times2
$((a \ \mathbf{mod}\ z) = 0)$
$\rightarrow \quad (((a \ \mathbf{mod}\ (y * z)) = (z * ((a \div z) \ \mathbf{mod}\ y)))$
$\quad \land \quad ((a \ \mathbf{mod}\ (z * y)) = (z * ((a \div z) \ \mathbf{mod}\ y))))$

THEOREM: remainder-times2-instance
$(((x * y) \ \mathbf{mod}\ (x * z)) = (x * (y \ \mathbf{mod}\ z)))$
$\land \quad (((x * z) \ \mathbf{mod}\ (y * z)) = ((x \ \mathbf{mod}\ y) * z))$

THEOREM: remainder-difference1
$((a \ \mathbf{mod}\ c) = (b \ \mathbf{mod}\ c))$
$\rightarrow \quad (((a - b) \ \mathbf{mod}\ c) = ((a \ \mathbf{mod}\ c) - (b \ \mathbf{mod}\ c)))$

DEFINITION:
double-remainder-induction $(a,\ b,\ c)$
$=$ **if** $c \simeq 0$ **then** 0
elseif $a < c$ **then** 0
elseif $b < c$ **then** 0
else double-remainder-induction $(a - c,\ b - c,\ c)$ **endif**

THEOREM: remainder-difference2
$(((a \ \mathbf{mod}\ c) = 0) \land ((b \ \mathbf{mod}\ c) \neq 0))$
$\rightarrow \quad (((a - b) \ \mathbf{mod}\ c)$
$\quad = \quad$ **if** $b < a$ **then** $c - (b \ \mathbf{mod}\ c)$
else 0 **endif**)

THEOREM: remainder-difference3
$$(((b \ \mathbf{mod}\ c) = \texttt{0}) \land ((a \ \mathbf{mod}\ c) \neq \texttt{0}))$$
$$\rightarrow \quad (((a - b) \ \mathbf{mod}\ c) = \mathbf{if}\ b < a \ \mathbf{then}\ a \ \mathbf{mod}\ c \ \mathbf{else}\ \texttt{0}\ \mathbf{endif})$$

EVENT: Disable remainder-difference3.

THEOREM: equal-remainder-difference-0
$$(((a - b) \ \mathbf{mod}\ c) = \texttt{0})$$
$$= \quad \mathbf{if}\ b \leq a \ \mathbf{then}\ (a \ \mathbf{mod}\ c) = (b \ \mathbf{mod}\ c) \ \mathbf{else}\ \mathbf{t}\ \mathbf{endif}$$

EVENT: Disable equal-remainder-difference-0.

THEOREM: lessp-plus-fact
$$(((b \ \mathbf{mod}\ x) = \texttt{0}) \land ((c \ \mathbf{mod}\ x) = \texttt{0}) \land (b < c) \land (a < x))$$
$$\rightarrow \quad (((a + b) < c) = \mathbf{t})$$

EVENT: Disable lessp-plus-fact.

THEOREM: remainder-plus-fact
$$(((b \ \mathbf{mod}\ x) = \texttt{0}) \land ((c \ \mathbf{mod}\ x) = \texttt{0}) \land (a < x))$$
$$\rightarrow \quad (((a + b) \ \mathbf{mod}\ c) = (a + (b \ \mathbf{mod}\ c)))$$

THEOREM: remainder-plus-times-times-proof
$$(a < b)$$
$$\rightarrow \quad (((a + (b * c)) \ \mathbf{mod}\ (b * d)) = (a + ((b * c) \ \mathbf{mod}\ (b * d))))$$

THEOREM: remainder-plus-times-times
$$(a < b)$$
$$\rightarrow \quad ((((a + (b * c)) \ \mathbf{mod}\ (b * d)) = (a + ((b * c) \ \mathbf{mod}\ (b * d)))) \land (((a + (c * b)) \ \mathbf{mod}\ (d * b)) = (a + ((c * b) \ \mathbf{mod}\ (d * b)))))$$

```
; REMAINDER-PLUS-TIMES-TIMES-INSTANCE is the completion of the rules
; TIMES-DISTRIBUTES-OVER-PLUS, REMAINDER-TIMES-TIMES and REMAINDER-PLUS-TIMES-TIMES
```

THEOREM: remainder-plus-times-times-instance
$$(a < b)$$
$$\rightarrow \quad ((((a + (b * c) + (b * d)) \ \mathbf{mod}\ (b * e))$$

$$= \quad (a + (b * ((c + d) \textbf{ mod } e))))$$
$$\wedge \quad (((a + (c * b) + (d * b)) \textbf{ mod } (e * b))$$
$$= \quad (a + (b * ((c + d) \textbf{ mod } e)))))$$

Theorem: remainder-remainder
$((b \textbf{ mod } a) = \mathtt{0}) \rightarrow (((n \textbf{ mod } b) \textbf{ mod } a) = (n \textbf{ mod } a))$

Theorem: remainder-1-arg1
$(\mathtt{1} \textbf{ mod } x)$
$= \quad \textbf{if } x = \mathtt{1} \textbf{ then } \mathtt{0}$
$\quad\quad \textbf{else } \mathtt{1} \textbf{ endif}$

Theorem: remainder-1-arg2
$(y \textbf{ mod } \mathtt{1}) = \mathtt{0}$

Theorem: remainder-x-x
$(x \textbf{ mod } x) = \mathtt{0}$

Theorem: transitivity-of-divides
$(((a \textbf{ mod } b) = \mathtt{0}) \wedge ((b \textbf{ mod } c) = \mathtt{0})) \rightarrow ((a \textbf{ mod } c) = \mathtt{0})$

```
; ---------- QUOTIENT, DIVIDES  ----------
```

Theorem: quotient-noop
$(b = \mathtt{1}) \rightarrow ((a \div b) = \mathrm{fix}\,(a))$

Theorem: quotient-of-non-number
$(a \notin \mathbf{N}) \rightarrow ((a \div n) = (\mathtt{0} \div n))$

Theorem: quotient-zero
$(x \simeq \mathtt{0}) \rightarrow ((y \div x) = \mathtt{0})$

Theorem: quotient-add1
$((a \textbf{ mod } b) = \mathtt{0})$
$\rightarrow \quad (((1 + a) \div b)$
$\quad\quad = \quad \textbf{if } b = \mathtt{1} \textbf{ then } 1 + (a \div b)$
$\quad\quad\quad\quad \textbf{else } a \div b \textbf{ endif})$

Theorem: equal-quotient-0
$((a \div b) = \mathtt{0}) = ((b \simeq \mathtt{0}) \vee (a < b))$

Theorem: quotient-sub1
$((a \not\simeq \mathtt{0}) \wedge (b \not\simeq \mathtt{0}))$
$\rightarrow \quad (((a - 1) \div b)$
$\quad\quad = \quad \textbf{if } (a \textbf{ mod } b) = \mathtt{0} \textbf{ then } (a \div b) - 1$
$\quad\quad\quad\quad \textbf{else } a \div b \textbf{ endif})$

THEOREM: quotient-plus-proof
$((b \mathbf{\ mod\ } c) = 0) \rightarrow (((a + b) \div c) = ((a \div c) + (b \div c)))$

THEOREM: quotient-plus
$$\begin{aligned}
&((a \mathbf{\ mod\ } c) = 0) \\
\rightarrow\ &((((a + b) \div c) = ((a \div c) + (b \div c))) \\
&\quad \land\ (((b + a) \div c) = ((a \div c) + (b \div c))) \\
&\quad \land\ (((x + y + a) \div c) \\
&\qquad = (((x + y) \div c) + (a \div c))))
\end{aligned}$$

```
; I need QUOTIENT-TIMES-INSTANCE to prove the more general QUOTIENT-TIMES,
; but I want QUOTIENT-TIMES-INSTANCE to be tried first (i.e. come after
; QUOTIENT-TIMES in the event list.) So first, prove QUOTIENT-TIMES-INSTANCE-TEMP,
; then prove QUOTIENT-TIMES, and finally give QUOTIENT-TIMES-INSTANCE.
```

THEOREM: quotient-times-instance-temp-proof
$$\begin{aligned}
&((y * x) \div y) \\
=\ &\mathbf{if}\ y \simeq 0\ \mathbf{then}\ 0 \\
&\mathbf{else}\ \mathrm{fix}\,(x)\ \mathbf{endif}
\end{aligned}$$

THEOREM: quotient-times-instance-temp
$$\begin{aligned}
&(((y * x) \div y) \\
&\ =\ \mathbf{if}\ y \simeq 0\ \mathbf{then}\ 0 \\
&\qquad \mathbf{else}\ \mathrm{fix}\,(x)\ \mathbf{endif}) \\
\land\ &(((x * y) \div y) \\
&\ =\ \mathbf{if}\ y \simeq 0\ \mathbf{then}\ 0 \\
&\qquad \mathbf{else}\ \mathrm{fix}\,(x)\ \mathbf{endif})
\end{aligned}$$

EVENT: Disable quotient-times-instance-temp.

THEOREM: quotient-times-proof
$((a \mathbf{\ mod\ } c) = 0) \rightarrow (((a * b) \div c) = (b * (a \div c)))$

THEOREM: quotient-times
$$\begin{aligned}
&((a \mathbf{\ mod\ } c) = 0) \\
\rightarrow\ &((((a * b) \div c) = (b * (a \div c))) \\
&\quad \land\ (((b * a) \div c) = (b * (a \div c))))
\end{aligned}$$

THEOREM: quotient-times-instance
$$\begin{aligned}
&(((y * x) \div y) \\
&\ =\ \mathbf{if}\ y \simeq 0\ \mathbf{then}\ 0 \\
&\qquad \mathbf{else}\ \mathrm{fix}\,(x)\ \mathbf{endif}) \\
\land\ &(((x * y) \div y) \\
&\ =\ \mathbf{if}\ y \simeq 0\ \mathbf{then}\ 0 \\
&\qquad \mathbf{else}\ \mathrm{fix}\,(x)\ \mathbf{endif})
\end{aligned}$$

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

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

EVENT: Disable quotient-times-times.

THEOREM: quotient-difference1
$((a \textbf{ mod } c) = (b \textbf{ mod } c))$
$\rightarrow \;\; (((a - b) \div c) = ((a \div c) - (b \div c)))$

THEOREM: quotient-lessp-arg1
$(a < b) \rightarrow ((a \div b) = \mathtt{0})$

THEOREM: quotient-difference2
$(((a \textbf{ mod } c) = \mathtt{0}) \land ((b \textbf{ mod } c) \neq \mathtt{0}))$
$\rightarrow \;\; (((a - b) \div c)$
$\qquad = \;\; \textbf{if } b < a \textbf{ then } (a \div c) - (1 + (b \div c))$
$\qquad\quad\;\; \textbf{else } \mathtt{0} \textbf{ endif})$

THEOREM: quotient-difference3
$(((b \textbf{ mod } c) = \mathtt{0}) \land ((a \textbf{ mod } c) \neq \mathtt{0}))$
$\rightarrow \;\; (((a - b) \div c)$
$\qquad = \;\; \textbf{if } b < a \textbf{ then } (a \div c) - (b \div c)$
$\qquad\quad\;\; \textbf{else } \mathtt{0} \textbf{ endif})$

THEOREM: remainder-equals-its-first-argument
$(a = (a \textbf{ mod } b)) = ((a \in \mathbf{N}) \land ((b \simeq \mathtt{0}) \lor (a < b)))$

EVENT: Disable remainder-equals-its-first-argument.

THEOREM: quotient-remainder-times
$((x \textbf{ mod } (a * b)) \div a) = ((x \div a) \textbf{ mod } b)$

THEOREM: quotient-remainder
$((c \textbf{ mod } a) = \mathtt{0}) \rightarrow (((b \textbf{ mod } c) \div a) = ((b \div a) \textbf{ mod } (c \div a)))$

THEOREM: quotient-remainder-instance
$((x \textbf{ mod } (a * b)) \div a) = ((x \div a) \textbf{ mod } b)$

THEOREM: quotient-plus-fact
$(((b \textbf{ mod } x) = 0) \wedge ((c \textbf{ mod } x) = 0) \wedge (a < x))$
$\rightarrow \quad (((a + b) \div c) = (b \div c))$

THEOREM: quotient-plus-times-times-proof
$(a < b)$
$\rightarrow \quad (((a + (b * c)) \div (b * d)) = ((b * c) \div (b * d)))$

THEOREM: quotient-plus-times-times
$(a < b)$
$\rightarrow \quad ((((a + (b * c)) \div (b * d)) = ((b * c) \div (b * d)))$
$\quad\quad \wedge \quad (((a + (b * c)) \div (b * d))$
$\quad\quad\quad\quad = \quad ((b * c) \div (b * d))))$

```
; QUOTIENT-PLUS-TIMES-TIMES-INSTANCE is the completion of the rules
; QUOTIENT-TIMES-TIMES, QUOTIENT-PLUS-TIMES-TIMES and TIMES-DISTRIBUTES-OVER-PLUS
```

THEOREM: quotient-plus-times-times-instance
$(a < b)$
$\rightarrow \quad ((((a + (b * c) + (b * d)) \div (b * e))$
$\quad\quad = \quad \textbf{if } b \simeq 0 \textbf{ then } 0$
$\quad\quad\quad\quad \textbf{else } (c + d) \div e \textbf{ endif})$
$\quad\quad \wedge \quad (((a + (c * b) + (d * b)) \div (e * b))$
$\quad\quad\quad\quad = \quad \textbf{if } b \simeq 0 \textbf{ then } 0$
$\quad\quad\quad\quad\quad\quad \textbf{else } (d + c) \div e \textbf{ endif}))$

THEOREM: quotient-quotient
$((b \div a) \div c) = (b \div (a * c))$

THEOREM: leq-quotient
$(a < b) \rightarrow ((a \div c) \leq (b \div c))$

THEOREM: quotient-1-arg2
$(n \div 1) = \text{fix}(n)$

THEOREM: quotient-1-arg1-casesplit
$(n \simeq 0) \vee (n = 1) \vee (1 < n)$

THEOREM: quotient-1-arg1
$(1 \div n)$
$= \quad \textbf{if } n = 1 \textbf{ then } 1$
$\quad\quad \textbf{else } 0 \textbf{ endif}$

THEOREM: quotient-x-x
$(x \not\simeq 0) \rightarrow ((x \div x) = 1)$

THEOREM: lessp-quotient
$((i \div j) < i) = ((i \not\simeq 0) \wedge (j \neq 1))$

```
;; Metalemma to cancel quotient-times expressions

;; ex.
;; (quotient (times a b) (times c (times d a)))  ->
;;                                              (if (not (zerop a))
;;                                                  (quotient (fix b) (times c d))
;;                                               (zero))
;;
```

DEFINITION:
cancel-quotient-times (x)
$=$ **if** (car (x) = `'quotient`)
  $\wedge$ (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 (`'quotient`,
     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)))))),
    `'(zero)`)
  else x **endif**
 else x **endif**

THEOREM: zerop-makes-quotient-zero-bridge
((car (x) = `'times`)
 $\wedge$ (car (y) = `'times`)
 $\wedge$ ($\neg$ eval\$ (**t**,
    and-not-zerop-tree (bagint (times-fringe (x), times-fringe (y))),
    a)))
$\rightarrow$ (((eval\$ (**t**, cadr (x), a) $*$ eval\$ (**t**, caddr (x), a))
  $\div$ (eval\$ (**t**, cadr (y), a) $*$ eval\$ (**t**, caddr (y), a)))
  $=$ 0)

EVENT: Disable zerop-makes-quotient-zero-bridge.

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

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

EVENT: Disable eval\$-quotient-times-tree-bagdiff.

THEOREM: correctness-of-cancel-quotient-times

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

```
;;; exp, log, and gcd
```

DEFINITION:

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

DEFINITION:

$\text{log}(base, n)$
$= \ $ **if** $base < \mathtt{2}$ **then** $\mathtt{0}$
elseif $n \simeq \mathtt{0}$ **then** $\mathtt{0}$
else $1 + \text{log}(base, n \div base)$ **endif**

DEFINITION:

$\text{gcd}(x, y)$
$= \ $ **if** $x \simeq \mathtt{0}$ **then** $\text{fix}(y)$
elseif $y \simeq \mathtt{0}$ **then** x
elseif $x < y$ **then** $\text{gcd}(x, y - x)$
else $\text{gcd}(x - y, y)$ **endif**

THEOREM: remainder-exp

$$(k \not\simeq \mathtt{0}) \rightarrow ((\text{exp}(n, k) \ \mathbf{mod} \ n) = \mathtt{0})$$

DEFINITION:

$\text{double-number-induction}(i, j)$
$= \ $ **if** $i \simeq \mathtt{0}$ **then** $\mathtt{0}$
elseif $j \simeq \mathtt{0}$ **then** $\mathtt{0}$
else $\text{double-number-induction}(i - 1, j - 1)$ **endif**

THEOREM: remainder-exp-exp

$$(i \le j) \rightarrow ((\text{exp}(a, j) \ \mathbf{mod} \ \text{exp}(a, i)) = \mathtt{0})$$

THEOREM: quotient-exp

$$
\begin{aligned}
&(k \not\simeq \mathtt{0}) \\
\rightarrow\quad & ((\exp(n,\ k) \div n) \\
&\quad = \ \ \textbf{if}\ n \simeq \mathtt{0}\ \ \textbf{then}\ \mathtt{0} \\
&\qquad\quad\ \textbf{else}\ \exp(n,\ k-1)\ \textbf{endif})
\end{aligned}
$$

THEOREM: exp-zero

$(k \simeq \mathtt{0}) \rightarrow (\exp(n,\ k) = \mathtt{1})$

THEOREM: exp-add1

$\exp(n,\ 1+k) = (n * \exp(n,\ k))$

THEOREM: exp-plus

$\exp(i,\ j+k) = (\exp(i,\ j) * \exp(i,\ k))$

THEOREM: exp-0-arg1

$$
\begin{aligned}
&\exp(\mathtt{0},\ k) \\
= \quad & \textbf{if}\ k \simeq \mathtt{0}\ \ \textbf{then}\ \mathtt{1} \\
& \textbf{else}\ \mathtt{0}\ \textbf{endif}
\end{aligned}
$$

THEOREM: exp-1-arg1

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

THEOREM: exp-0-arg2

$\exp(n,\ \mathtt{0}) = \mathtt{1}$

THEOREM: exp-times

$\exp(i * j,\ k) = (\exp(i,\ k) * \exp(j,\ k))$

THEOREM: exp-exp

$\exp(\exp(i,\ j),\ k) = \exp(i,\ j * k)$

THEOREM: equal-exp-0

$(\exp(n,\ k) = \mathtt{0}) = ((n \simeq \mathtt{0}) \wedge (k \not\simeq \mathtt{0}))$

THEOREM: equal-exp-1

$$
\begin{aligned}
&(\exp(n,\ k) = \mathtt{1}) \\
= \quad & \textbf{if}\ k \simeq \mathtt{0}\ \ \textbf{then}\ \mathbf{t} \\
& \textbf{else}\ n = \mathtt{1}\ \textbf{endif}
\end{aligned}
$$

THEOREM: exp-difference

$((c \le b) \wedge (a \not\simeq \mathtt{0})) \rightarrow (\exp(a,\ b-c) = (\exp(a,\ b) \div \exp(a,\ c)))$

THEOREM: equal-log-0

$(\log(\mathit{base},\ n) = \mathtt{0}) = ((\mathit{base} < \mathtt{2}) \vee (n \simeq \mathtt{0}))$

THEOREM: log-0
$(n \simeq \mathtt{0}) \rightarrow (\log(base,\ n) = \mathtt{0})$

THEOREM: log-1
$(\mathtt{1} < base) \rightarrow (\log(base,\ \mathtt{1}) = \mathtt{1})$

DEFINITION:
double-log-induction $(base,\ a,\ b)$
$=$ **if** $base < \mathtt{2}$ **then** $\mathtt{0}$
elseif $a \simeq \mathtt{0}$ **then** $\mathtt{0}$
elseif $b \simeq \mathtt{0}$ **then** $\mathtt{0}$
else double-log-induction $(base,\ a \div base,\ b \div base)$ **endif**

THEOREM: leq-log-log
$(n \leq m) \rightarrow (\log(c,\ n) \leq \log(c,\ m))$

THEOREM: log-quotient
$(\mathtt{1} < c) \rightarrow (\log(c,\ n \div c) = (\log(c,\ n) - 1))$

THEOREM: log-quotient-times-proof
$(\mathtt{1} < c) \rightarrow (\log(c,\ n \div (c * m)) = (\log(c,\ n \div m) - 1))$

THEOREM: log-quotient-times
$(\mathtt{1} < c)$
$\rightarrow \quad ((\log(c,\ n \div (c * m)) = (\log(c,\ n \div m) - 1))$
$\quad \wedge \quad (\log(c,\ n \div (m * c)) = (\log(c,\ n \div m) - 1)))$

THEOREM: log-quotient-exp
$(\mathtt{1} < c) \rightarrow (\log(c,\ n \div \exp(c,\ m)) = (\log(c,\ n) - m))$

THEOREM: log-times-proof
$((\mathtt{1} < c) \wedge (n \not\simeq \mathtt{0})) \rightarrow (\log(c,\ c * n) = (1 + \log(c,\ n)))$

THEOREM: log-times
$((\mathtt{1} < c) \wedge (n \not\simeq \mathtt{0}))$
$\rightarrow \quad ((\log(c,\ c * n) = (1 + \log(c,\ n)))$
$\quad \wedge \quad (\log(c,\ n * c) = (1 + \log(c,\ n))))$

THEOREM: log-times-exp-proof
$((\mathtt{1} < c) \wedge (n \not\simeq \mathtt{0})) \rightarrow (\log(c,\ n * \exp(c,\ m)) = (\log(c,\ n) + m))$

THEOREM: log-times-exp
$((\mathtt{1} < c) \wedge (n \not\simeq \mathtt{0}))$
$\rightarrow \quad ((\log(c,\ n * \exp(c,\ m)) = (\log(c,\ n) + m))$
$\quad \wedge \quad (\log(c,\ \exp(c,\ m) * n) = (\log(c,\ n) + m)))$

THEOREM: log-exp
$(1 < c) \rightarrow (\log(c, \exp(c, n)) = (1 + n))$

THEOREM: commutativity-of-gcd
$\mathrm{gcd}(b, a) = \mathrm{gcd}(a, b)$

DEFINITION:
single-number-induction (n)
$=$ **if** $n \simeq 0$ **then** 0
else single-number-induction $(n - 1)$ **endif**

THEOREM: gcd-0
$(\mathrm{gcd}(0, x) = \mathrm{fix}(x)) \wedge (\mathrm{gcd}(x, 0) = \mathrm{fix}(x))$

THEOREM: gcd-1
$(\mathrm{gcd}(1, x) = 1) \wedge (\mathrm{gcd}(x, 1) = 1)$

THEOREM: equal-gcd-0
$(\mathrm{gcd}(a, b) = 0) = ((a \simeq 0) \wedge (b \simeq 0))$

THEOREM: lessp-gcd
$(b \not\simeq 0) \rightarrow (((b < \mathrm{gcd}(a, b)) = \mathbf{f}) \wedge ((b < \mathrm{gcd}(b, a)) = \mathbf{f}))$

THEOREM: gcd-plus-instance-temp-proof
$\mathrm{gcd}(a, a + b) = \mathrm{gcd}(a, b)$

THEOREM: gcd-plus-instance-temp
$(\mathrm{gcd}(a, a + b) = \mathrm{gcd}(a, b)) \wedge (\mathrm{gcd}(a, b + a) = \mathrm{gcd}(a, b))$

THEOREM: gcd-plus-proof
$((b \ \mathbf{mod}\ a) = 0) \rightarrow (\mathrm{gcd}(a, b + c) = \mathrm{gcd}(a, c))$

THEOREM: gcd-plus
$$\begin{aligned}
&((b \ \mathbf{mod}\ a) = 0) \\
\rightarrow\ & ((\mathrm{gcd}(a, b + c) = \mathrm{gcd}(a, c)) \\
&\quad \wedge\ (\mathrm{gcd}(a, c + b) = \mathrm{gcd}(a, c)) \\
&\quad \wedge\ (\mathrm{gcd}(b + c, a) = \mathrm{gcd}(a, c)) \\
&\quad \wedge\ (\mathrm{gcd}(c + b, a) = \mathrm{gcd}(a, c)))
\end{aligned}$$

THEOREM: gcd-plus-instance
$(\mathrm{gcd}(a, a + b) = \mathrm{gcd}(a, b)) \wedge (\mathrm{gcd}(a, b + a) = \mathrm{gcd}(a, b))$

THEOREM: remainder-gcd
$((a \ \mathbf{mod}\ \mathrm{gcd}(a, b)) = 0) \wedge ((b \ \mathbf{mod}\ \mathrm{gcd}(a, b)) = 0)$

THEOREM: distributivity-of-times-over-gcd-proof
$\mathrm{gcd}(x * z, y * z) = (z * \mathrm{gcd}(x, y))$

Theorem: distributivity-of-times-over-gcd
$(\text{gcd}\,(x * z,\ y * z) = (z * \text{gcd}\,(x,\ y)))$
$\wedge\quad (\text{gcd}\,(z * x,\ y * z) = (z * \text{gcd}\,(x,\ y)))$
$\wedge\quad (\text{gcd}\,(x * z,\ z * y) = (z * \text{gcd}\,(x,\ y)))$
$\wedge\quad (\text{gcd}\,(z * x,\ z * y) = (z * \text{gcd}\,(x,\ y)))$

Theorem: gcd-is-the-greatest
$((x \not\simeq \mathtt{0}) \wedge (y \not\simeq \mathtt{0}) \wedge ((x \ \mathbf{mod}\ z) = \mathtt{0}) \wedge ((y \ \mathbf{mod}\ z) = \mathtt{0}))$
$\rightarrow\quad (z \leq \text{gcd}\,(x,\ y))$

Theorem: common-divisor-divides-gcd
$(((x \ \mathbf{mod}\ z) = \mathtt{0}) \wedge ((y \ \mathbf{mod}\ z) = \mathtt{0})) \rightarrow ((\text{gcd}\,(x,\ y) \ \mathbf{mod}\ z) = \mathtt{0})$

```
; We prove ASSOCIATIVITY-OF-GCD and COMMUTATIVITY2-OF-GCD roughly the same way.
; Use GCD-IS-THE-GREATEST twice to show that each side of the equality is
; less than or equal to the other side.
```

Theorem: associativity-of-gcd-zero-case
$((a \simeq \mathtt{0}) \vee (b \simeq \mathtt{0}) \vee (c \simeq \mathtt{0})) \rightarrow (\text{gcd}\,(\text{gcd}\,(a,\ b),\ c) = \text{gcd}\,(a,\ \text{gcd}\,(b,\ c)))$

Theorem: associativity-of-gcd
$\text{gcd}\,(\text{gcd}\,(a,\ b),\ c) = \text{gcd}\,(a,\ \text{gcd}\,(b,\ c))$

Theorem: commutativity2-of-gcd-zero-case
$((a \simeq \mathtt{0}) \vee (b \simeq \mathtt{0}) \vee (c \simeq \mathtt{0})) \rightarrow (\text{gcd}\,(b,\ \text{gcd}\,(a,\ c)) = \text{gcd}\,(a,\ \text{gcd}\,(b,\ c)))$

Theorem: commutativity2-of-gcd
$\text{gcd}\,(b,\ \text{gcd}\,(a,\ c)) = \text{gcd}\,(a,\ \text{gcd}\,(b,\ c))$

Theorem: gcd-x-x
$\text{gcd}\,(x,\ x) = \text{fix}\,(x)$

Theorem: gcd-idempotence
$(\text{gcd}\,(x,\ \text{gcd}\,(x,\ y)) = \text{gcd}\,(x,\ y)) \wedge (\text{gcd}\,(y,\ \text{gcd}\,(x,\ y)) = \text{gcd}\,(x,\ y))$

```
;;; new stuff
```

Theorem: gcd-zero
$(x \simeq \mathtt{0}) \rightarrow ((\text{gcd}\,(x,\ y) = \text{fix}\,(y)) \wedge (\text{gcd}\,(y,\ x) = \text{fix}\,(y)))$

Theorem: remainder-quotient-gcd-0
$((x \ \mathbf{mod}\ a) = \mathtt{0}) \rightarrow ((x \ \mathbf{mod}\ (x \div a)) = \mathtt{0})$

Theorem: remainder-gcd-0
$((x \ \mathbf{mod}\ a) = \mathtt{0})$
$\rightarrow\quad (((x \ \mathbf{mod}\ \text{gcd}\,(a,\ b)) = \mathtt{0}) \wedge ((x \ \mathbf{mod}\ \text{gcd}\,(b,\ a)) = \mathtt{0}))$

```
;I think this is not important
;(prove-lemma gcd-of-reduce-numbers-helper-helper (rewrite)
;            (equal
;             (remainder a (gcd (quotient a (gcd a b)) (quotient b (gcd a b))))
;             0))

; I believe that the following works just as well as this one because
; of commutativity-of-gcd
;(prove-lemma gcd-of-reduced-numbers-helper (rewrite)
;       (implies
;       (not (zerop a))
;       (and
;        (equal (gcd (quotient a (gcd a b))
;                    (quotient b (gcd a b)))
;             1)
;        (equal (gcd (quotient b (gcd a b))
;                    (quotient a (gcd a b)))
;             1)
;        (equal (gcd (quotient b (gcd b a))
;                    (quotient a (gcd b a)))
;             1)
;        (equal (gcd (quotient a (gcd b a))
;                    (quotient b (gcd b a)))
;             1))))
```

THEOREM: gcd-quotient-gcd

$(a \not\simeq 0)$

$\rightarrow \quad ((\text{gcd}\,(a \div \text{gcd}\,(a,\, b),\, b \div \text{gcd}\,(a,\, b)) = 1)$

$\qquad \wedge \quad (\text{gcd}\,(b \div \text{gcd}\,(b,\, a),\, a \div \text{gcd}\,(b,\, a)) = 1))$

THEOREM: lessp-gcd2

$((b < \text{gcd}\,(a,\, b)) = ((b \simeq 0) \wedge (a \not\simeq 0)))$

$\wedge \quad ((b < \text{gcd}\,(b,\, a)) = ((b \simeq 0) \wedge (a \not\simeq 0)))$

THEOREM: gcd-times-proof

$\text{gcd}\,(y,\, x * y) = \text{fix}\,(y)$

THEOREM: gcd-times

$(\text{gcd}\,(y,\, x * y) = \text{fix}\,(y)) \wedge (\text{gcd}\,(y,\, y * x) = \text{fix}\,(y))$

```
;; replaced by gcd-zero
;(lemma gcd-zerop (rewrite)
;       (implies
;       (zerop x)
```

```
;       (and
;        (equal (gcd x z) (fix z))
;        (equal (gcd z x) (fix z))))
;       ((enable-theory naturals)
;       (enable gcd)))

; (disable gcd-zerop) ; subsumed by gcd-remainder
```

THEOREM: gcd-remainder
$((x \textbf{ mod } z) = 0) \rightarrow ((\text{gcd}\,(x,\, z) = \text{fix}\,(z)) \wedge (\text{gcd}\,(z,\, x) = \text{fix}\,(z)))$

EVENT: Disable gcd-zero.

```
; subsumed by gcd-remainder

; no longer needed
;(lemma equal-times-x-x-fact (rewrite)
;       (implies (and (not (equal x 0))
;                     (not (equal y 1)))
;                (and
;                 (equal (equal (times x y) x) f)
;                 (equal (equal (times y x) x) f)))
;       ((enable-theory naturals)
;       (induct (times y x))))
```

THEOREM: equal-times-x-x
$$(((x * y) = x) = (((x \in \mathbf{N}) \wedge (y = 1)) \vee (x = 0)))$$
$$\wedge \quad (((y * x) = x) = (((x \in \mathbf{N}) \wedge (y = 1)) \vee (x = 0)))$$

THEOREM: equal-x-gcd-x-y
$$(y \in \mathbf{N})$$
$$\rightarrow \quad (((x = \text{gcd}\,(x,\, y)) = ((x \in \mathbf{N}) \wedge ((y \textbf{ mod } x) = 0)))$$
$$\wedge \quad ((x = \text{gcd}\,(y,\, x)) = ((x \in \mathbf{N}) \wedge ((y \textbf{ mod } x) = 0))))$$

THEOREM: quotient-difference
$$((x - y) \div z)$$
$$= \quad \textbf{if } (x \textbf{ mod } z) < (y \textbf{ mod } z) \textbf{ then } ((x \div z) - (y \div z)) - 1$$
$$\textbf{else } (x \div z) - (y \div z) \textbf{ endif}$$

THEOREM: equal-as-remainder
$$(((x \textbf{ mod } y) = 0) \wedge ((y \textbf{ mod } x) = 0) \wedge (x \in \mathbf{N}) \wedge (y \in \mathbf{N}))$$
$$\rightarrow \quad ((x = y) = \mathbf{t})$$

EVENT: Disable equal-as-remainder.

```
;; add so as to control rewriting equal better
```

THEOREM: equal-gcd-gcd-as-remainder

$(((\mathrm{gcd}\,(x,\ y)\ \mathbf{mod}\ \mathrm{gcd}\,(a,\ b)) = \mathtt{0}) \land ((\mathrm{gcd}\,(a,\ b)\ \mathbf{mod}\ \mathrm{gcd}\,(x,\ y)) = \mathtt{0}))$
$\rightarrow \quad ((\mathrm{gcd}\,(a,\ b) = \mathrm{gcd}\,(x,\ y)) = \mathbf{t})$

THEOREM: remainder-gcd-0-means

$(((\mathrm{gcd}\,(a,\ b)\ \mathbf{mod}\ c) = \mathtt{0}) \rightarrow ((b\ \mathbf{mod}\ c) = \mathtt{0}))$
$\land \quad (((\mathrm{gcd}\,(b,\ a)\ \mathbf{mod}\ c) = \mathtt{0}) \rightarrow ((b\ \mathbf{mod}\ c) = \mathtt{0}))$

THEOREM: remainder-gcd-arg1

$((\mathrm{gcd}\,(a,\ b)\ \mathbf{mod}\ c) = \mathtt{0})$
$= \quad \mathbf{if}\ (a\ \mathbf{mod}\ c) = \mathtt{0}$
$\quad\quad \mathbf{then\ if}\ (b\ \mathbf{mod}\ c) = \mathtt{0}\ \mathbf{then\ t}$
$\quad\quad\quad\quad \mathbf{else\ f\ endif}$
$\quad\quad \mathbf{else\ f\ endif}$

THEOREM: remainder-diff1

$(((x\ \mathbf{mod}\ d) = \mathtt{0}) \land ((w * x) \not< z))$
$\rightarrow \quad (((((w * x) - z)\ \mathbf{mod}\ d) = \mathtt{0}) = ((z\ \mathbf{mod}\ d) = \mathtt{0}))$

THEOREM: remainder-diff2

$(((x\ \mathbf{mod}\ d) = \mathtt{0})$
$\land \quad ((w * x) \not< z)$
$\land \quad ((((w * x) - z)\ \mathbf{mod}\ d) = \mathtt{0}))$
$\rightarrow \quad ((z\ \mathbf{mod}\ d) = \mathtt{0})$

THEOREM: remainder-gcd-difference-times-hack

$((w * x) \not< z) \rightarrow ((z\ \mathbf{mod}\ \mathrm{gcd}\,(x,\ (w * x) - z)) = \mathtt{0})$

THEOREM: gcd-difference1

$((y\ \mathbf{mod}\ x) = \mathtt{0})$
$\rightarrow \quad (\mathrm{gcd}\,(x,\ y - z)$
$\quad\quad = \quad \mathbf{if}\ y < z\ \mathbf{then}\ \mathrm{fix}\,(x)$
$\quad\quad\quad\quad \mathbf{else}\ \mathrm{gcd}\,(x,\ z)\ \mathbf{endif})$

THEOREM: gcd-difference2

$((y\ \mathbf{mod}\ x) = \mathtt{0})$
$\rightarrow \quad (\mathrm{gcd}\,(x,\ z - y)$
$\quad\quad = \quad \mathbf{if}\ z < y\ \mathbf{then}\ \mathrm{fix}\,(x)$
$\quad\quad\quad\quad \mathbf{else}\ \mathrm{gcd}\,(x,\ z)\ \mathbf{endif})$

EVENT: Let us define the theory *addition* to consist of the following events: equal-plus-0, equal-difference-0, commutativity-of-plus, commutativity2-of-plus, plus-zero-arg2, plus-add1-arg2, plus-add1-arg1, associativity-of-plus, plus-difference-arg1, plus-difference-arg2, diff-diff-arg1, diff-diff-arg2, correctness-of-cancel-equal-plus, correctness-of-cancel-difference-plus, difference-elim, difference-leq-arg1, difference-add1-arg2, difference-sub1-arg2, difference-difference-arg1, difference-difference-arg2, difference-x-x, correctness-of-cancel-lessp-plus.

EVENT: Let us define the theory *multiplication* to consist of the following events: equal-times-0, equal-times-1, equal-sub1-0, times-zero, times-add1, commutativity-of-times, times-distributes-over-plus, commutativity2-of-times, associativity-of-times, times-distributes-over-difference, times-quotient, times-1-arg1, lessp-times1, lessp-times2, lessp-times3, lessp-plus-times1, lessp-plus-times2, lessp-1-times, correctness-of-cancel-lessp-times, correctness-of-cancel-equal-times, equal-times-x-x.

EVENT: Let us define the theory *remainders* to consist of the following events: lessp-remainder, remainder-noop, remainder-of-non-number, remainder-zero, remainder-quotient-elim, remainder-add1, remainder-plus, equal-remainder-plus-0, remainder-times1, remainder-times1-instance, remainder-times2, remainder-times2-instance, remainder-difference1, remainder-difference2, remainder-plus-times-times, remainder-plus-times-times-instance, remainder-remainder, remainder-1-arg1, remainder-1-arg2, remainder-x-x, remainder-quotient-gcd-0, remainder-gcd-0, remainder-gcd-arg1, remainder-diff1, remainder-diff2.

EVENT: Let us define the theory *quotients* to consist of the following events: quotient-noop, quotient-of-non-number, quotient-zero, quotient-add1, equal-quotient-0, quotient-sub1, quotient-plus, quotient-times, quotient-times-instance, quotient-difference1, quotient-lessp-arg1, quotient-difference2, quotient-difference3, quotient-remainder-times, quotient-remainder, quotient-remainder-instance, quotient-plus-times-times, quotient-plus-times-times-instance, quotient-quotient, quotient-1-arg2, quotient-1-arg1, quotient-x-x, lessp-quotient, correctness-of-cancel-quotient-times, quotient-difference.

EVENT: Let us define the theory *exponentiation* to consist of the following events: equal-exp-0, equal-exp-1, exp-exp, exp-add1, exp-times, exp-1-arg1, exp-zero, exp-0-arg2, exp-0-arg1, exp-difference, exp-plus, quotient-exp, remainder-exp-exp, remainder-exp.

EVENT: Let us define the theory *logs* to consist of the following events: log-exp, log-times-exp, log-times, log-quotient-exp, log-quotient-times, log-quotient, log-

1, log-0, equal-log-0, exp-exp.

EVENT: Let us define the theory *gcds* to consist of the following events: commutativity2-of-gcd, associativity-of-gcd, common-divisor-divides-gcd, distributivity-of-times-over-gcd, lessp-gcd, equal-gcd-0, gcd-idempotence, gcd-x-x, remainder-gcd, gcd-plus, gcd-plus-instance, gcd-1, commutativity-of-gcd, gcd-zero, lessp-gcd2, gcd-times, gcd-remainder, equal-x-gcd-x-y, equal-gcd-gcd-as-remainder, remainder-gcd-0-means, gcd-quotient-gcd, gcd-difference1, gcd-difference2.

EVENT: Let us define the theory *naturals* to consist of the following events: addition, multiplication, remainders, quotients, exponentiation, logs, gcds.

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

Index