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

DEFINITION:
delete $(x,\ l)$
$=$ **if** listp (l)
then if $x = \mathrm{car}(l)$ **then** cdr (l)
else cons (car (l), delete $(x$, cdr $(l)))$ **endif**
else l **endif**

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

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

DEFINITION:
occurrences $(x,\ l)$
$=$ **if** listp (l)
then if $x = \mathrm{car}(l)$ **then** $1 +$ occurrences $(x$, cdr $(l))$
else occurrences $(x$, cdr $(l))$ **endif**
else 0 **endif**

DEFINITION:
permutation $(a,\ b)$
$=$ **if** listp (a)
then (car $(a) \in b) \wedge$ permutation (cdr (a), delete (car $(a),\ b))$
else $\neg$ listp (b) **endif**

DEFINITION:
setp (l)
$=$ **if** listp (l) **then** (car $(l) \notin$ cdr $(l)) \wedge$ setp (cdr $(l))$
else t endif

DEFINITION:
subbagp $(x,\ y)$
$=$ **if** listp (x)

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

THEOREM: listp-delete
listp (delete $(x,\ l)$)
$=$ **if** listp (l) **then** $(x \neq$ car $(l)) \vee$ listp (cdr (l))
else f endif

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

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

THEOREM: equal-occurrences-zero
(occurrences $(x,\ l) = \mathtt{0}) = (x \notin l)$

THEOREM: occurrences-in-a-set
setp (l)
$\rightarrow$ (occurrences $(x,\ l)$
$=$ **if** $x \in l$ **then** $\mathtt{1}$
else $\mathtt{0}$ **endif**)

THEOREM: occurrences-cons
occurrences $(x,$ cons $(y,\ l))$
$=$ **if** $x = y$ **then** $1 +$ occurrences $(x,\ l)$
else occurrences $(x,\ l)$ **endif**

THEOREM: occurrences-delete
occurrences $(x,$ delete $(y,\ l))$
$=$ **if** $x = y$
then if $x \in l$ **then** occurrences $(x,\ l) - 1$
else $\mathtt{0}$ **endif**
else occurrences $(x,\ l)$ **endif**

THEOREM: occurrences-bagdiff
occurrences $(x,$ bagdiff $(a,\ b)) =$ (occurrences $(x,\ a) -$ occurrences $(x,\ b)$)

THEOREM: occurrences-bagint
occurrences $(x,$ bagint $(a,\ b))$
$=$ **if** occurrences $(x,\ a) <$ occurrences $(x,\ b)$ **then** occurrences $(x,\ a)$
else occurrences $(x,\ b)$ **endif**

THEOREM: member-non-list
$(\neg$ listp $(l)) \rightarrow (x \notin l)$

THEOREM: member-cons
$(x \in \text{cons}\,(y,\ l)) = ((x = y) \lor (x \in l))$

THEOREM: member-delete-implies-membership
$(x \in \text{delete}\,(y,\ l)) \to (x \in l)$

THEOREM: member-delete
$(x \in \text{delete}\,(y,\ l))$
$= \quad \textbf{if } x \in l$
$\quad\quad \textbf{then if } x = y \textbf{ then } \texttt{1} < \text{occurrences}\,(x,\ l)$
$\quad\quad\quad\quad \textbf{else t endif}$
$\quad\quad \textbf{else f endif}$

THEOREM: permutation-implies-membership-equivalence
$\text{permutation}\,(a,\ b) \to ((x \in a) = (x \in b))$

THEOREM: member-bagdiff
$(x \in \text{bagdiff}\,(a,\ b)) = (\text{occurrences}\,(x,\ b) < \text{occurrences}\,(x,\ a))$

THEOREM: member-bagint
$(x \in \text{bagint}\,(a,\ b)) = ((x \in a) \land (x \in b))$

THEOREM: not-permutation
$(\text{listp}\,(b) \land (\text{car}\,(b) \notin a)) \to (\neg\ \text{permutation}\,(a,\ b))$

THEOREM: permutation-right-cons1
$(\text{listp}\,(b) \land (\text{car}\,(b) \in a))$
$\to \quad (\text{permutation}\,(a,\ b) = \text{permutation}\,(\text{delete}\,(\text{car}\,(b),\ a),\ \text{cdr}\,(b)))$

THEOREM: permutation-right-cons
$\text{permutation}\,(a,\ \text{cons}\,(x,\ b))$
$= \quad \textbf{if } x \in a \textbf{ then } \text{permutation}\,(\text{delete}\,(x,\ a),\ b)$
$\quad\quad \textbf{else f endif}$

THEOREM: commutativity-of-permutation
$\text{permutation}\,(b,\ a) = \text{permutation}\,(a,\ b)$

THEOREM: permutation-reflexivity
$\text{permutation}\,(l,\ l)$

THEOREM: permutation-delete-delete
$(\text{permutation}\,(a,\ b) \land (x \in a) \land (x \in b))$
$\to \quad \text{permutation}\,(\text{delete}\,(x,\ a),\ \text{delete}\,(x,\ b))$

DEFINITION:
transitivity-of-permutation-induction $(a,\ b,\ c)$

$$
\begin{aligned}
= \quad & \textbf{if } \mathrm{listp}\,(a) \\
& \textbf{then } \text{transitivity-of-permutation-induction}\,(\mathrm{cdr}\,(a), \\
& \qquad\qquad \mathrm{delete}\,(\mathrm{car}\,(a),\ b), \\
& \qquad\qquad \mathrm{delete}\,(\mathrm{car}\,(a),\ c)) \\
& \textbf{else } \mathtt{0} \textbf{ endif}
\end{aligned}
$$

THEOREM: transitivity-of-permutation-base-case
$((a \simeq \textbf{nil}) \land \mathrm{permutation}\,(a,\ b) \land \mathrm{permutation}\,(b,\ c)) \rightarrow \mathrm{permutation}\,(a,\ c)$

THEOREM: transitivity-of-permutation-induction-step

$$
\begin{aligned}
& (\mathrm{listp}\,(a) \\
& \ \land \quad ((\mathrm{permutation}\,(\mathrm{cdr}\,(a),\ \mathrm{delete}\,(\mathrm{car}\,(a),\ b)) \\
& \qquad\quad \land \quad \mathrm{permutation}\,(\mathrm{delete}\,(\mathrm{car}\,(a),\ b),\ \mathrm{delete}\,(\mathrm{car}\,(a),\ c))) \\
& \qquad \rightarrow \quad \mathrm{permutation}\,(\mathrm{cdr}\,(a),\ \mathrm{delete}\,(\mathrm{car}\,(a),\ c))) \\
& \ \land \quad \mathrm{permutation}\,(a,\ b) \\
& \ \land \quad \mathrm{permutation}\,(b,\ c)) \\
& \rightarrow \quad \mathrm{permutation}\,(a,\ c)
\end{aligned}
$$

THEOREM: transitivity-of-permutation
$(\mathrm{permutation}\,(a,\ b) \land \mathrm{permutation}\,(b,\ c)) \rightarrow \mathrm{permutation}\,(a,\ c)$

THEOREM: setp-delete
$\mathrm{setp}\,(a) \rightarrow \mathrm{setp}\,(\mathrm{delete}\,(x,\ a))$

DEFINITION:
setp-permutation-induction $(a,\ b)$

$$
\begin{aligned}
= \quad & \textbf{if } \mathrm{listp}\,(a) \ \textbf{ then } \text{setp-permutation-induction}\,(\mathrm{cdr}\,(a),\ \mathrm{delete}\,(\mathrm{car}\,(a),\ b)) \\
& \textbf{else } \mathtt{0} \textbf{ endif}
\end{aligned}
$$

THEOREM: setp-permutation-base-case
$(\neg\ \mathrm{listp}\,(a)) \rightarrow \mathrm{setp}\,(a)$

THEOREM: setp-permutation-induction-step

$$
\begin{aligned}
& (\mathrm{listp}\,(a) \\
& \ \land \quad \mathrm{setp}\,(b) \\
& \ \land \quad \mathrm{permutation}\,(a,\ b) \\
& \ \land \quad ((\mathrm{setp}\,(\mathrm{delete}\,(\mathrm{car}\,(a),\ b)) \land \mathrm{permutation}\,(\mathrm{cdr}\,(a),\ \mathrm{delete}\,(\mathrm{car}\,(a),\ b))) \\
& \qquad \rightarrow \quad \mathrm{setp}\,(\mathrm{cdr}\,(a)))) \\
& \rightarrow \quad \mathrm{setp}\,(a)
\end{aligned}
$$

THEOREM: setp-permutation
$(\mathrm{setp}\,(b) \land \mathrm{permutation}\,(a,\ b)) \rightarrow \mathrm{setp}\,(a)$

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

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
$\text{subbagp}\,(\text{bagint}\,(x,\ y),\ y)$

EVENT: Let us define the theory *sets-and-bags* to consist of the following events: subbagp-bagint2, subbagp-bagint1, subbagp-cdr2, subbagp-cdr1, subbagp-delete, setp-delete, permutation-delete-delete, permutation-reflexivity, commutativity-of-permutation, permutation-right-cons, permutation-right-cons1, not-permutation, member-bagint, member-bagdiff, member-delete, member-delete-implies-membership, member-cons, member-non-list, occurrences-bagint, occurrences-bagdiff, occurrences-delete, occurrences-cons, occurrences-in-a-set, equal-occurrences-zero, delete-delete, delete-non-member, listp-delete.

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

THEOREM: plus-stopper
$(a \simeq 0) \rightarrow ((a + b) = \text{fix}\,(b))$

THEOREM: plus-right-id2
$(y \simeq 0) \rightarrow ((x + y) = \text{fix}\,(x))$

THEOREM: plus-add1
$$\begin{array}{l} (x + (1 + y)) \\ = \quad \textbf{if } y \in \mathbf{N} \textbf{ then } 1 + (x + y) \\ \qquad \textbf{else } 1 + x \textbf{ endif} \end{array}$$

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

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

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

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

THEOREM: equal-plus-1

$((a + b) = 1)$
$= \quad (((a \simeq 0) \wedge (b = 1)) \vee ((a = 1) \wedge (b \simeq 0)))$

THEOREM: plus-cancellation

$((a + b) = (a + c)) = (\mathrm{fix}\,(b) = \mathrm{fix}\,(c))$

THEOREM: plus-difference-arg1-casesplit

$((a - b) + c)$
$= \quad$ **if** $b < a$ **then** $(a + c) - b$
else $\mathrm{fix}\,(c)$ **endif**

THEOREM: plus-difference-arg1

$(b < a) \rightarrow (((a - b) + c) = ((a + c) - b))$

THEOREM: plus-difference-arg2-casesplit

$(a + (b - c))$
$= \quad$ **if** $c < b$ **then** $(a + b) - c$
else $\mathrm{fix}\,(a)$ **endif**

THEOREM: plus-difference-arg2

$(c < b) \rightarrow ((a + (b - c)) = ((a + b) - c))$

THEOREM: plus-difference-difference-casesplit

$((a - b) + (c - d))$
$= \quad$ **if if** $a < b$ **then f**
else t endif
then if if $c < d$ **then f**
else t endif then $(a + c) - (b + d)$
else $a - b$ **endif**
elseif if $c < d$ **then f**
else t endif then $c - d$
else 0 **endif**

THEOREM: plus-difference-difference

$((b < a) \wedge (d < c))$
$\rightarrow \quad (((a - b) + (c - d)) = ((a + c) - (b + d)))$

THEOREM: difference-elim

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

THEOREM: difference-leq-arg1

if $b < a$ **then f**
else t endif
$\rightarrow \quad ((a - b) = 0)$

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

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

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

THEOREM: difference-plus-cancellation1
$((x + y) - (x + z)) = (y - z)$

THEOREM: difference-plus-cancellation2
$((a + c) - (b + c)) = (a - b)$

THEOREM: difference-cancellation-0
$(x = (x - y)) = ((x \in \mathbf{N}) \land ((x = \mathtt{0}) \lor (y \simeq \mathtt{0})))$

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

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

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

THEOREM: difference-add1-plus-cancellation
$((1 + (y + z)) - z) = (1 + y)$

THEOREM: difference-sub1-arg2
$(a - (b - 1))$
$=$ **if if** $\mathtt{1} < b$ **then f**
else t endif then $\mathrm{fix}\,(a)$
elseif $a < b$ **then** $\mathtt{0}$
else $1 + (a - b)$ **endif**

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

THEOREM: difference-difference-arg2
$(a - (b - c))$
$=$ **if if** $b < c$ **then f**
else t endif then $(a + c) - b$
else $\mathrm{fix}\,(a)$ **endif**

THEOREM: difference-difference-difference

$$(\textbf{if } a < b \textbf{ then f} \; \textbf{else t endif} \wedge \textbf{if } c < d \textbf{ then f} \; \textbf{else t endif}) \rightarrow (((a - b) - (c - d)) = ((a + d) - (b + c)))$$

THEOREM: difference-plus-cancellation3

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

DEFINITION:

plus-fringe (x)

$$= \textbf{if } \text{listp}(x) \wedge (\text{car}(x) = \texttt{'plus}) \; \textbf{then } \text{append}(\text{plus-fringe}(\text{cadr}(x)), \text{plus-fringe}(\text{caddr}(x))) \; \textbf{else } \text{list}(x) \textbf{ endif}$$

DEFINITION:

plus-tree (l)

$$= \textbf{if } l \simeq \textbf{nil} \textbf{ then } \texttt{''0} \; \textbf{elseif } \text{cdr}(l) \simeq \textbf{nil} \textbf{ then } \text{list}(\texttt{'fix}, \text{car}(l)) \; \textbf{elseif } \text{cddr}(l) \simeq \textbf{nil} \textbf{ then } \text{list}(\texttt{'plus}, \text{car}(l), \text{cadr}(l)) \; \textbf{else } \text{list}(\texttt{'plus}, \text{car}(l), \text{plus-tree}(\text{cdr}(l))) \textbf{ endif}$$

THEOREM: numberp-eval\$-plus

$$(\text{listp}(x) \wedge (\text{car}(x) = \texttt{'plus})) \rightarrow (\text{eval\$}(\mathbf{t}, x, a) \in \mathbf{N})$$

THEOREM: numberp-eval\$-plus-tree

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

THEOREM: member-implies-plus-tree-greatereqp

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

THEOREM: plus-tree-delete

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

THEOREM: subbagp-implies-plus-tree-greatereqp

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

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

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

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
$$((c \in \text{plus-fringe}\,(x)) \wedge (\text{eval\$}\,(\mathbf{t},\ c,\ a) \in \mathbf{N})) \to (\text{eval\$}\,(\mathbf{t},\ x,\ a) \in \mathbf{N})$$

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

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

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

DEFINITION:
$$
\begin{array}{l}
\text{cancel-equal-plus}\,(x) \\
= \quad \textbf{if}\ \text{listp}\,(x) \wedge (\text{car}\,(x) = \texttt{'equal}) \\
\qquad \textbf{then if}\ \text{listp}\,(\text{cadr}\,(x)) \\
\qquad\qquad\qquad \wedge\quad (\text{caadr}\,(x) = \texttt{'plus}) \\
\qquad\qquad\qquad \wedge\quad \text{listp}\,(\text{caddr}\,(x)) \\
\qquad\qquad\qquad \wedge\quad (\text{caaddr}\,(x) = \texttt{'plus}) \\
\qquad\qquad \textbf{then}\ \text{list}\,(\texttt{'equal}, \\
\qquad\qquad\qquad\qquad \text{plus-tree}\,(\text{bagdiff}\,(\text{plus-fringe}\,(\text{cadr}\,(x)), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{bagint}\,(\text{plus-fringe}\,(\text{cadr}\,(x)), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{plus-fringe}\,(\text{caddr}\,(x))))), \\
\qquad\qquad\qquad\qquad \text{plus-tree}\,(\text{bagdiff}\,(\text{plus-fringe}\,(\text{caddr}\,(x)), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{bagint}\,(\text{plus-fringe}\,(\text{cadr}\,(x)), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{plus-fringe}\,(\text{caddr}\,(x)))))) \\
\qquad\qquad \textbf{elseif}\ \text{listp}\,(\text{cadr}\,(x)) \\
\qquad\qquad\qquad\quad \wedge\quad (\text{caadr}\,(x) = \texttt{'plus}) \\
\qquad\qquad\qquad\quad \wedge\quad (\text{caddr}\,(x) \in \text{plus-fringe}\,(\text{cadr}\,(x))) \\
\qquad\qquad \textbf{then}\ \text{list}\,(\texttt{'if},
\end{array}
$$

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

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

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\$ ($\mathbf{t}$, x, a) = eval\$ ($\mathbf{t}$, cancel-difference-plus (x), a)

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

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

THEOREM: diff-diff-arg2
$(c < b) \rightarrow ((a - (b - c)) = ((a + c) - b))$

THEOREM: diff-diff-diff-casesplit
$((a - b) - (c - d))$
$=$ **if** $b < a$
then if $d < c$ **then** $(a + d) - (b + c)$
else $a - b$ **endif**
else 0 **endif**

THEOREM: diff-diff-diff
$((b < a) \wedge (d < c))$
$\rightarrow (((a - b) - (c - d)) = ((a + d) - (b + c)))$

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

THEOREM: diff-add1-arg2
$(b < a) \rightarrow ((a - (1 + b)) = ((a - b) - 1))$

THEOREM: diff-sub1-arg2-casesplit
$(a - (b - 1))$
$=$ **if if** 1 $< b$ **then f**
else t endif then fix (a)
elseif $a < b$ **then** 0
else $1 + (a - b)$ **endif**

THEOREM: diff-sub1-arg2
(**if** $b <$ 1 **then f**
else t endif
$\wedge$ **if** $a < b$ **then f**
else t endif)
$\rightarrow ((a - (b - 1)) = (1 + (a - b)))$

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

THEOREM: lessp-difference-cancellation
$((a - c) < (b - c))$
$= \ \ \textbf{if if } a < c \textbf{ then f}$
$\quad\quad \textbf{else t endif then } a < b$
$\quad \textbf{else } c < b \textbf{ endif}$

THEOREM: lessp-difference
$((x \not\simeq \mathtt{0}) \land (y \not\simeq \mathtt{0})) \rightarrow (((x - y) < x) = \mathbf{t})$

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

EVENT: Enable cancel-lessp-plus; name this event ‘cancel-lessp-plus-off’.

EVENT: Enable lessp-difference-cancellation; name this event ‘lessp-difference-cancellation-off’.

EVENT: Enable cancel-difference-plus; name this event 'cancel-difference-plus-off'.

EVENT: Enable cancel-equal-plus; name this event 'cancel-equal-plus-off'.

EVENT: Enable listp-eval$; name this event 'listp-eval$-off'.

EVENT: Enable eval$-quote; name this event 'eval$-quote-off'.

EVENT: Enable cadr-eval$-list; name this event 'cadr-eval$-list-off'.

EVENT: Enable member-implies-numberp; name this event 'member-implies-numberp-off'.

EVENT: Enable plus-tree-plus-fringe; name this event 'plus-tree-plus-fringe-off'.

EVENT: Enable eval$-plus-tree-append; name this event 'eval$-plus-tree-append-off'.

EVENT: Enable bridge-to-subbagp-implies-plus-tree-greatereqp; name this event 'bridge-to-subbagp-implies-plus-tree-greatereqp-off'.

EVENT: Enable numberp-eval$-bridge; name this event 'numberp-eval$-bridge-off'.

EVENT: Enable plus-tree-bagdiff; name this event 'plus-tree-bagdiff-off'.

EVENT: Enable subbagp-implies-plus-tree-greatereqp; name this event 'subbagp-implies-plus-tree-greatereqp-off'.

EVENT: Enable plus-tree-delete; name this event 'plus-tree-delete-off'.

EVENT: Enable member-implies-plus-tree-greatereqp; name this event 'member-implies-plus-tree-greatereqp-off'.

EVENT: Enable numberp-eval$-plus-tree; name this event 'numberp-eval$-plus-tree-off'.

EVENT: Enable numberp-eval$-plus; name this event 'numberp-eval$-plus-off'.

EVENT: Enable difference-plus-cancellation3; name this event 'difference-plus-cancellation3-off'.

EVENT: Enable difference-difference-difference; name this event 'difference-difference-difference-off'.

EVENT: Enable difference-difference-arg2; name this event 'difference-difference-arg2-off'.

EVENT: Enable difference-difference-arg1; name this event 'difference-difference-arg1-off'.

EVENT: Enable difference-sub1-arg2; name this event 'difference-sub1-arg2-off'.

EVENT: Enable difference-add1-plus-cancellation; name this event 'difference-add1-plus-cancellation-off'.

EVENT: Enable difference-2; name this event 'difference-2-off'.

EVENT: Enable difference-1; name this event 'difference-1-off'.

EVENT: Enable difference-cancellation-1; name this event 'difference-cancellation-1-off'.

EVENT: Enable difference-cancellation-0; name this event 'difference-cancellation-0-off'.

EVENT: Enable difference-plus-cancellation2; name this event 'difference-plus-cancellation2-off'.

EVENT: Enable difference-plus-cancellation1; name this event 'difference-plus-cancellation1-off'.

EVENT: Enable difference-plus; name this event 'difference-plus-off'.

EVENT: Enable difference-x-x; name this event 'difference-x-x-off'.

EVENT: Enable plus-cancellation; name this event 'plus-cancellation-off'.

EVENT: Enable open-plus; name this event 'open-plus-off'.

EVENT: Let us define the theory *addition* to consist of the following events: plus-stopper, plus-right-id2, plus-add1, commutativity2-of-plus, commutativity-of-plus, associativity-of-plus, equal-plus-0, equal-plus-1, plus-difference-arg1-casesplit, plus-difference-arg2-casesplit, plus-difference-difference-casesplit, plus-difference-arg1, plus-difference-arg2, plus-difference-difference, correctness-of-cancel-equal-plus, difference-elim, difference-leq-arg1, equal-difference-0, correctness-of-cancel-difference-plus, diff-diff-arg1, diff-diff-arg2-casesplit, diff-diff-diff-casesplit, diff-add1-arg2-casesplit, diff-sub1-arg2-casesplit, diff-diff-arg2, diff-diff-diff, diff-add1-arg2, diff-sub1-arg2, diff-x-x, lessp-difference, correctness-of-cancel-lessp-plus.

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

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

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

THEOREM: times-sub1
$(b \not\simeq \mathtt{0}) \rightarrow ((a * (b - 1)) = ((a * b) - a))$

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

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
$(((a - b) * c) = ((a * c) - (b * c)))$
$\wedge \quad ((a * (b - c)) = ((a * b) - (a * c)))$

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

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

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

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

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

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

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

THEOREM: lessp-times2-proof
(**if** $b < a$ **then f**
else t endif
$\wedge \quad (c \not\simeq 0))$
$\rightarrow \quad (((b * c) < a) = \mathbf{f})$

THEOREM: lessp-times2
(**if** $b < a$ **then f**
else t endif
$\wedge \quad (c \not\simeq 0))$
$\rightarrow \quad ((((b * c) < a) = \mathbf{f}) \wedge (((c * b) < a) = \mathbf{f}))$

THEOREM: lessp-times3-proof1
$((a \not\simeq 0) \wedge (1 < b)) \rightarrow (a < (a * b))$

Theorem: lessp-times3-proof2
$(a < (a * b)) \rightarrow ((a \not\simeq \mathtt{0}) \wedge (\mathtt{1} < b))$

Theorem: lessp-times3
$((a < (a * b)) = ((a \not\simeq \mathtt{0}) \wedge (\mathtt{1} < b)))$
$\wedge \quad ((a < (b * a)) = ((a \not\simeq \mathtt{0}) \wedge (\mathtt{1} < b)))$

Theorem: lessp-times-cancellation-proof
$((x * z) < (y * z)) = ((z \not\simeq \mathtt{0}) \wedge (x < y))$

Theorem: lessp-times-cancellation1
$(((x * z) < (y * z)) = ((z \not\simeq \mathtt{0}) \wedge (x < y)))$
$\wedge \quad (((z * x) < (y * z)) = ((z \not\simeq \mathtt{0}) \wedge (x < y)))$
$\wedge \quad (((x * z) < (z * y)) = ((z \not\simeq \mathtt{0}) \wedge (x < y)))$
$\wedge \quad (((z * x) < (z * y)) = ((z \not\simeq \mathtt{0}) \wedge (x < y)))$

Theorem: lessp-times-cancellation2
$(((b * a * c) < (a * d)) = ((a \not\simeq \mathtt{0}) \wedge ((b * c) < d)))$
$\wedge \quad ((((a * c) * b) < (a * d))$
$\qquad = \quad ((a \not\simeq \mathtt{0}) \wedge ((b * c) < d)))$
$\wedge \quad (((b * a * c) < (d * a)) = ((a \not\simeq \mathtt{0}) \wedge ((b * c) < d)))$
$\wedge \quad ((((a * c) * b) < (d * a))$
$\qquad = \quad ((a \not\simeq \mathtt{0}) \wedge ((b * c) < d)))$
$\wedge \quad (((b * c * a) < (a * d)) = ((a \not\simeq \mathtt{0}) \wedge ((b * c) < d)))$
$\wedge \quad ((((c * a) * b) < (a * d))$
$\qquad = \quad ((a \not\simeq \mathtt{0}) \wedge ((b * c) < d)))$
$\wedge \quad (((b * c * a) < (d * a)) = ((a \not\simeq \mathtt{0}) \wedge ((b * c) < d)))$
$\wedge \quad ((((c * a) * b) < (d * a))$
$\qquad = \quad ((a \not\simeq \mathtt{0}) \wedge ((b * c) < d)))$

Theorem: lessp-plus-times-proof
$(x < a) \rightarrow (((x + (a * b)) < (a * c)) = (b < c))$

Theorem: lessp-plus-times1
$(((a + (b * c)) < b) = ((a < b) \wedge (c \simeq \mathtt{0})))$
$\wedge \quad (((a + (c * b)) < b) = ((a < b) \wedge (c \simeq \mathtt{0})))$
$\wedge \quad ((((c * b) + a) < b) = ((a < b) \wedge (c \simeq \mathtt{0})))$
$\wedge \quad ((((b * c) + a) < b) = ((a < b) \wedge (c \simeq \mathtt{0})))$

Theorem: lessp-plus-times2
$((a \not\simeq \mathtt{0}) \wedge (x < a))$
$\rightarrow \quad ((((x + (a * b)) < (a * c)) = (b < c))$
$\qquad \wedge \quad (((x + (b * a)) < (a * c)) = (b < c))$
$\qquad \wedge \quad (((x + (a * b)) < (c * a)) = (b < c))$
$\qquad \wedge \quad (((x + (b * a)) < (c * a)) = (b < c)))$

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

THEOREM: lessp-1-times
$(1 < (a * b))$
$= \quad (\neg ((a \simeq 0) \vee (b \simeq 0) \vee ((a = 1) \wedge (b = 1))))$

THEOREM: equal-sub1-times-0
$(((a * b) - 1) = 0)$
$= \quad ((a \simeq 0) \vee (b \simeq 0) \vee ((a = 1) \wedge (b = 1)))$

EVENT: Let us define the theory *multiplication* to consist of the following events: equal-sub1-times-0, lessp-1-times, lessp-plus-times2, lessp-plus-times1, lessp-times-cancellation2, lessp-times-cancellation1, lessp-times3, lessp-times2, lessp-times1, times-1-arg1, equal-times-1, equal-times-0, times-quotient, times-distributes-over-difference, associativity-of-times, commutativity2-of-times, times-distributes-over-plus, commutativity-of-times, times-sub1, times-add1, times-zero.

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

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

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

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

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

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

THEOREM: remainder-sub1
$((a \not\simeq 0) \wedge (b \not\simeq 0))$
$\rightarrow \quad (((a - 1) \text{ **mod** } b)$
$\qquad = \quad \textbf{if } (a \textbf{ mod } b) = 0 \textbf{ then } b - 1$
$\qquad\qquad \textbf{else } (a \textbf{ mod } b) - 1 \textbf{ endif})$

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
$$\begin{array}{l} ((a \textbf{ mod } c) = 0) \\ \rightarrow \quad ((((a + b) \textbf{ mod } c) = (b \textbf{ mod } c)) \\ \qquad \wedge \quad (((b + a) \textbf{ mod } c) = (b \textbf{ mod } c)) \\ \qquad \wedge \quad (((x + y + a) \textbf{ mod } c) = ((x + y) \textbf{ mod } c))) \end{array}$$

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
$$\begin{array}{l} ((a \textbf{ mod } c) = 0) \\ \rightarrow \quad (((((a + b) \textbf{ mod } c) = 0) = ((b \textbf{ mod } c) = 0)) \\ \qquad \wedge \quad ((((b + a) \textbf{ mod } c) = 0) = ((b \textbf{ mod } c) = 0)) \\ \qquad \wedge \quad ((((x + y + a) \textbf{ mod } c) = 0) \\ \qquad \qquad = \quad (((x + y) \textbf{ mod } c) = 0))) \end{array}$$

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
$$\begin{array}{l} (a < c) \\ \rightarrow \quad (((((a + b) \textbf{ mod } c) = (b \textbf{ mod } c)) = (a \simeq 0)) \\ \qquad \wedge \quad ((((b + a) \textbf{ mod } c) = (b \textbf{ mod } c)) = (a \simeq 0))) \end{array}$$

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

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

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

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

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

THEOREM: remainder-times-times

$$(((x * y) \textbf{ mod } (x * z)) = (x * (y \textbf{ mod } z))) \\ \wedge \quad (((x * z) \textbf{ mod } (y * z)) = ((x \textbf{ mod } y) * z))$$

THEOREM: remainder-times2-proof

$$((a \textbf{ mod } z) = 0) \rightarrow ((a \textbf{ mod } (z * y)) = (z * ((a \div z) \textbf{ mod } y)))$$

THEOREM: remainder-times2

$$((a \textbf{ mod } z) = 0) \\ \rightarrow \quad (((a \textbf{ mod } (y * z)) = (z * ((a \div z) \textbf{ mod } y))) \\ \wedge \quad ((a \textbf{ mod } (z * y)) = (z * ((a \div z) \textbf{ mod } y))))$$

THEOREM: remainder-times2-instance

$$(((x * y) \textbf{ mod } (x * z)) = (x * (y \textbf{ mod } z))) \\ \wedge \quad (((x * z) \textbf{ mod } (y * z)) = ((x \textbf{ mod } y) * z))$$

THEOREM: remainder-difference1

$$((a \textbf{ mod } c) = (b \textbf{ mod } c)) \\ \rightarrow \quad (((a - b) \textbf{ mod } c) = ((a \textbf{ mod } c) - (b \textbf{ mod } c)))$$

DEFINITION:

double-remainder-induction $(a,\ b,\ c)$

$$\begin{aligned} = \quad & \textbf{if } c \simeq 0 \textbf{ then } 0 \\ & \textbf{elseif } a < c \textbf{ then } 0 \\ & \textbf{elseif } b < c \textbf{ then } 0 \\ & \textbf{else } \text{double-remainder-induction}\,(a - c,\ b - c,\ c) \textbf{ endif} \end{aligned}$$

THEOREM: remainder-difference2

$$(((a \textbf{ mod } c) = 0) \wedge ((b \textbf{ mod } c) \neq 0)) \\ \rightarrow \quad (((a - b) \textbf{ mod } c) \\ = \quad \textbf{if } b < a \textbf{ then } c - (b \textbf{ mod } c) \\ \textbf{else } 0 \textbf{ endif})$$

THEOREM: remainder-difference3

$$(((b \textbf{ mod } c) = 0) \wedge ((a \textbf{ mod } c) \neq 0)) \\ \rightarrow \quad (((a - b) \textbf{ mod } c) \\ = \quad \textbf{if } b < a \textbf{ then } a \textbf{ mod } c \\ \textbf{else } 0 \textbf{ endif})$$

THEOREM: equal-remainder-difference-0

$$(((a - b) \textbf{ mod } c) = 0) \\ = \quad \textbf{if if } a < b \textbf{ then f} \\ \textbf{else t endif then } (a \textbf{ mod } c) = (b \textbf{ mod } c) \\ \textbf{else t endif}$$

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

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

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

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

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

THEOREM: remainder-exp
$(k \not\simeq 0) \rightarrow ((\exp(n, k) \textbf{ mod } n) = 0)$

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

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

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

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

THEOREM: transitivity-of-divides
$(((a \textbf{ mod } b) = 0) \land ((b \textbf{ mod } c) = 0)) \rightarrow ((a \textbf{ mod } c) = 0)$

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

THEOREM: remainder-exp-exp
if $j < i$ **then f**
else t endif
$\rightarrow$ $((\exp(a, j)$ **mod** $\exp(a, i)) = \texttt{0})$

EVENT: Enable lessp-plus-fact; name this event 'lessp-plus-fact-off'.

EVENT: Enable equal-remainder-difference-0; name this event 'equal-remainder-difference-0-off'.

EVENT: Enable remainder-difference3; name this event 'remainder-difference3-off'.

EVENT: Enable remainder-times-times; name this event 'remainder-times-times-off'.

EVENT: Enable equal-remainder-plus-remainder; name this event 'equal-remainder-plus-remainder-off'.

EVENT: Enable plus-remainder-times-quotient; name this event 'plus-remainder-times-quotient-off'.

EVENT: Let us define the theory *remainders* to consist of the following events: remainder-exp-exp, remainder-x-x, remainder-1-arg2, remainder-1-arg1, remainder-remainder, remainder-exp, remainder-plus-times-times-instance, remainder-plus-times-times, remainder-difference2, remainder-difference1, remainder-times2-instance, remainder-times2, remainder-times1-instance, remainder-times1, equal-remainder-plus-0, remainder-plus, remainder-add1, remainder-sub1, remainder-quotient-elim, remainder-zero, remainder-of-non-number, remainder-noop, lessp-remainder.

THEOREM: quotient-noop
$(b = \texttt{1}) \rightarrow ((a \div b) = \text{fix}(a))$

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

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

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

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

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

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

THEOREM: quotient-plus
$((a\ \mathbf{mod}\ c) = \mathtt{0})$
$\rightarrow \quad ((((a + b) \div c) = ((a \div c) + (b \div c)))$
$\quad \wedge \quad (((b + a) \div c) = ((a \div c) + (b \div c)))$
$\quad \wedge \quad (((x + y + a) \div c)$
$\quad\quad = \quad (((x + y) \div c) + (a \div c))))$

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

THEOREM: quotient-times-instance-temp
$(((y * x) \div y)$
$= \quad \mathbf{if}\ y \simeq \mathtt{0}\ \mathbf{then}\ \mathtt{0}$
$\quad \mathbf{else}\ \mathrm{fix}\,(x)\ \mathbf{endif})$
$\wedge \quad (((x * y) \div y)$
$\quad = \quad \mathbf{if}\ y \simeq \mathtt{0}\ \mathbf{then}\ \mathtt{0}$
$\quad\quad \mathbf{else}\ \mathrm{fix}\,(x)\ \mathbf{endif})$

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

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

THEOREM: quotient-times-instance

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

THEOREM: quotient-times-times-proof

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

THEOREM: quotient-times-times

$$(((x * y) \div (x * z)) = \textbf{if } x \simeq \mathtt{0} \textbf{ then } \mathtt{0} \textbf{ else } y \div z \textbf{ endif}) \wedge (((x * z) \div (y * z)) = \textbf{if } z \simeq \mathtt{0} \textbf{ then } \mathtt{0} \textbf{ else } x \div y \textbf{ endif})$$

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

$$(((a \textbf{ mod } c) = \mathtt{0}) \wedge ((b \textbf{ mod } c) \neq \mathtt{0})) \rightarrow (((a - b) \div c) = \textbf{if } b < a \textbf{ then } (a \div c) - (1 + (b \div c)) \textbf{ else } \mathtt{0} \textbf{ endif})$$

THEOREM: quotient-difference3

$$(((b \textbf{ mod } c) = \mathtt{0}) \wedge ((a \textbf{ mod } c) \neq \mathtt{0})) \rightarrow (((a - b) \div c) = \textbf{if } b < a \textbf{ then } (a \div c) - (b \div c) \textbf{ else } \mathtt{0} \textbf{ endif})$$

THEOREM: remainder-equals-its-first-argument

$$(a = (a \textbf{ mod } b)) = ((a \in \mathbf{N}) \wedge ((b \simeq \mathtt{0}) \vee (a < b)))$$

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
$(((b \textbf{ mod } x) = \mathtt{0}) \wedge ((c \textbf{ mod } x) = \mathtt{0}) \wedge (a < x))$
$\rightarrow \quad (((a + b) \div c) = (b \div c))$

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
$(a < b)$
$\rightarrow \quad ((((a + (b * c)) \div (b * d)) = ((b * c) \div (b * d)))$
$\quad \wedge \quad (((a + (b * c)) \div (b * d))$
$\quad\quad = \quad ((b * c) \div (b * d))))$

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

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

THEOREM: quotient-exp
$(k \not\simeq \mathtt{0})$
$\rightarrow \quad ((\exp(n, k) \div n)$
$\quad = \quad \textbf{if } n \simeq \mathtt{0} \textbf{ then } \mathtt{0}$
$\quad\quad \textbf{else } \exp(n, k - 1) \textbf{ endif})$

THEOREM: leq-quotient
$(a < b)$
$\rightarrow \quad \textbf{if } (b \div c) < (a \div c) \textbf{ then f}$
$\quad \textbf{else t endif}$

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

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

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

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

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

EVENT: Enable quotient-times-instance-temp; name this event 'quotient-times-instance-temp-off'.

EVENT: Enable remainder-equals-its-first-argument; name this event 'remainder-equals-its-first-argument-off'.

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

THEOREM: exp-zero
$(k \simeq 0) \rightarrow (\exp(n, k) = 1)$

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

THEOREM: exp-1-arg1
$\exp(1, k) = 1$

THEOREM: exp-0-arg2
$\exp(n, 0) = 1$

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

THEOREM: equal-exp-1
$(\exp(n, k) = 1)$
$= \quad$ **if** $k \simeq 0$ **then t**
else $n = 1$ **endif**

THEOREM: exp-difference
(**if** $b < c$ **then f**
else t endif
$\wedge \quad (a \not\simeq 0))$
$\rightarrow \quad (\exp(a, b - c) = (\exp(a, b) \div \exp(a, c)))$

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.

DEFINITION:
$\log(base, n)$
$= \quad$ **if** $base < 2$ **then** 0
elseif $n \simeq 0$ **then** 0
else $1 + \log(base, n \div base)$ **endif**

THEOREM: equal-log-0
$(\log(base, n) = 0) = ((base < 2) \vee (n \simeq 0))$

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

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

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

THEOREM: leq-log-log

$$\textbf{if } m < n \textbf{ then f} \\ \textbf{else t endif} \\ \rightarrow \quad \textbf{if } \log(c, m) < \log(c, n) \textbf{ then f} \\ \textbf{else t endif}$$

THEOREM: log-quotient

$$(1 < c) \rightarrow (\log(c, n \div c) = (\log(c, n) - 1))$$

THEOREM: log-quotient-times-proof

$$(1 < c) \rightarrow (\log(c, n \div (c * m)) = (\log(c, n \div m) - 1))$$

THEOREM: log-quotient-times

$$(1 < c) \\ \rightarrow \quad ((\log(c, n \div (c * m)) = (\log(c, n \div m) - 1)) \\ \wedge \quad (\log(c, n \div (m * c)) = (\log(c, n \div m) - 1)))$$

THEOREM: log-quotient-exp

$$(1 < c) \rightarrow (\log(c, n \div \exp(c, m)) = (\log(c, n) - m))$$

THEOREM: log-times-proof

$$((1 < c) \wedge (n \not\simeq 0)) \rightarrow (\log(c, c * n) = (1 + \log(c, n)))$$

THEOREM: log-times

$$((1 < c) \wedge (n \not\simeq 0)) \\ \rightarrow \quad ((\log(c, c * n) = (1 + \log(c, n))) \\ \wedge \quad (\log(c, n * c) = (1 + \log(c, n))))$$

THEOREM: log-times-exp-proof

$$((1 < c) \wedge (n \not\simeq 0)) \rightarrow (\log(c, n * \exp(c, m)) = (\log(c, n) + m))$$

THEOREM: log-times-exp

$$((1 < c) \wedge (n \not\simeq 0)) \\ \rightarrow \quad ((\log(c, n * \exp(c, m)) = (\log(c, n) + m)) \\ \wedge \quad (\log(c, \exp(c, m) * n) = (\log(c, n) + m)))$$

THEOREM: log-exp

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

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.

DEFINITION:
$\gcd(x, y)$

$$
\begin{array}{ll}
= & \textbf{if } x \simeq \texttt{0} \textbf{ then } \mathrm{fix}\,(y) \\
 & \textbf{elseif } y \simeq \texttt{0} \textbf{ then } x \\
 & \textbf{elseif } x < y \textbf{ then } \mathrm{gcd}\,(x,\ y - x) \\
 & \textbf{else } \mathrm{gcd}\,(x - y,\ y) \textbf{ endif}
\end{array}
$$

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

DEFINITION:
single-number-induction (n)

$$
\begin{array}{ll}
= & \textbf{if } n \simeq \texttt{0} \textbf{ then } \texttt{0} \\
 & \textbf{else } \text{single-number-induction}\,(n - 1) \textbf{ endif}
\end{array}
$$

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

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

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

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

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

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

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

THEOREM: gcd-plus

$$
\begin{array}{ll}
 & ((b \textbf{ mod } a) = \texttt{0}) \\
\to & ((\mathrm{gcd}\,(a,\ b + c) = \mathrm{gcd}\,(a,\ c)) \\
 & \quad \land \quad (\mathrm{gcd}\,(a,\ c + b) = \mathrm{gcd}\,(a,\ c)) \\
 & \quad \land \quad (\mathrm{gcd}\,(b + c,\ a) = \mathrm{gcd}\,(a,\ c)) \\
 & \quad \land \quad (\mathrm{gcd}\,(c + b,\ a) = \mathrm{gcd}\,(a,\ c)))
\end{array}
$$

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

THEOREM: remainder-gcd
$((a \textbf{ mod } \mathrm{gcd}\,(a,\ b)) = \texttt{0}) \land ((b \textbf{ mod } \mathrm{gcd}\,(a,\ b)) = \texttt{0})$

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

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

THEOREM: gcd-is-the-greatest
$((x \not\simeq 0) \land (y \not\simeq 0) \land ((x \ \mathbf{mod}\ z) = 0) \land ((y \ \mathbf{mod}\ z) = 0))$
$\rightarrow\quad \mathbf{if}\ \text{gcd}(x,\ y) < z\ \ \mathbf{then\ f}$
$\quad\quad \mathbf{else\ t\ endif}$

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

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

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

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

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

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-0, gcd-idempotence, gcd-x-x, remainder-gcd, gcd-plus, gcd-plus-instance, gcd-1, commutativity-of-gcd.

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

DEFINITION:
$\text{integerp}(x) = ((x \in \mathbf{N}) \lor (\text{negativep}(x) \land (\text{negative-guts}(x) \not\simeq 0)))$

Definition:
fix-int (x)
$=$ **if** integerp (x) **then** x
else `0` **endif**

Definition:
izerop (i)
$=$ **if** integerp (i) **then** $i = \texttt{0}$
else t endif

Definition:
ilessp $(i,\ j)$
$=$ **if** negativep (i)
then if negativep (j) **then** negative-guts (j) $<$ negative-guts (i)
else t endif
elseif negativep (j) **then f**
else $i < j$ **endif**

Definition: ileq $(i,\ j)$ $=$ (ilessp $(i,\ j)$ $\vee$ $(i = j)$)

Definition:
iplus $(x,\ y)$
$=$ **if** negativep (x)
then if negativep (y)
then if (negative-guts $(x) \simeq \texttt{0}$) $\wedge$ (negative-guts $(y) \simeq \texttt{0}$) **then** `0`
else $-$ (negative-guts (x) $+$ negative-guts (y)) **endif**
elseif $y <$ negative-guts (x) **then** $-$ (negative-guts $(x) - y$)
else $y -$ negative-guts (x) **endif**
elseif negativep (y)
then if $x <$ negative-guts (y) **then** $-$ (negative-guts $(y) - x$)
else $x -$ negative-guts (y) **endif**
else $x + y$ **endif**

Definition:
ineg (x)
$=$ **if** negativep (x) **then** negative-guts (x)
elseif $x \simeq \texttt{0}$ **then** `0`
else $-\ x$ **endif**

Definition: idifference $(x,\ y)$ $=$ iplus $(x,$ ineg $(y))$

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

DEFINITION:
itimes $(i,\ j)$
$=$ **if** negativep (i)
then if negativep (j) **then** negative-guts $(i) *$ negative-guts (j)
else fix-int $(-$ (negative-guts $(i) * j))$ **endif**
elseif negativep (j) **then** fix-int $(-\ (i *$ negative-guts $(j)))$
else $i * j$ **endif**

THEOREM: integerp-fix-int
integerp (fix-int (x))

THEOREM: integerp-iplus
integerp (iplus $(x,\ y)$)

THEOREM: integerp-idifference
integerp (idifference $(x,\ y)$)

THEOREM: integerp-ineg
integerp (ineg (x))

THEOREM: integerp-iabs
integerp (iabs (x))

THEOREM: integerp-itimes
integerp (itimes $(x,\ y)$)

THEOREM: fix-int-fix-int
fix-int (fix-int (x)) $=$ fix-int (x)

THEOREM: fix-int-iplus
fix-int (iplus $(a,\ b)$) $=$ iplus $(a,\ b)$

THEOREM: fix-int-idifference
fix-int (idifference $(a,\ b)$) $=$ idifference $(a,\ b)$

THEOREM: fix-int-ineg
fix-int (ineg (x)) $=$ ineg (x)

THEOREM: fix-int-iabs
fix-int (iabs (x)) $=$ iabs (x)

THEOREM: fix-int-itimes
fix-int (itimes $(x,\ y)$) $=$ itimes $(x,\ y)$

THEOREM: ineg-id
$(\neg$ integerp $(x)) \rightarrow$ (ineg $(x) = 0$)

THEOREM: ineg-iplus
$\text{ineg}(\text{iplus}(a,\ b)) = \text{iplus}(\text{ineg}(a),\ \text{ineg}(b))$

THEOREM: ineg-ineg
$\text{ineg}(\text{ineg}(x)) = \text{fix-int}(x)$

THEOREM: ineg-idifference
$\text{ineg}(\text{idifference}(a,\ b)) = \text{iplus}(\text{ineg}(a),\ b)$

THEOREM: ineg-fix-int
$\text{ineg}(\text{fix-int}(x)) = \text{ineg}(x)$

THEOREM: iplus-left-id
$(\neg\ \text{integerp}(x)) \rightarrow (\text{iplus}(x,\ y) = \text{fix-int}(y))$

THEOREM: iplus-right-id
$(\neg\ \text{integerp}(y)) \rightarrow (\text{iplus}(x,\ y) = \text{fix-int}(x))$

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

THEOREM: commutativity2-of-iplus
$\text{iplus}(x,\ \text{iplus}(y,\ z)) = \text{iplus}(y,\ \text{iplus}(x,\ z))$

THEOREM: commutativity-of-iplus
$\text{iplus}(x,\ y) = \text{iplus}(y,\ x)$

THEOREM: associativity-of-iplus
$\text{iplus}(\text{iplus}(x,\ y),\ z) = \text{iplus}(x,\ \text{iplus}(y,\ z))$

THEOREM: iplus-cancellation
$(\text{iplus}(a,\ b) = \text{iplus}(a,\ c)) = (\text{fix-int}(b) = \text{fix-int}(c))$

THEOREM: iplus-ineg1
$\text{iplus}(a,\ \text{ineg}(a)) = \mathtt{0}$

THEOREM: iplus-ineg2
$\text{iplus}(\text{ineg}(a),\ a) = \mathtt{0}$

THEOREM: iplus-fix-int1
$\text{iplus}(\text{fix-int}(a),\ b) = \text{iplus}(a,\ b)$

THEOREM: iplus-fix-int2
$\text{iplus}(a,\ \text{fix-int}(b)) = \text{iplus}(a,\ b)$

THEOREM: iplus-idifference-arg1
$\text{iplus}(\text{idifference}(a,\ b),\ c) = \text{idifference}(\text{iplus}(a,\ c),\ b)$

THEOREM: iplus-idifference-arg2
$\text{iplus}(a, \text{idifference}(b, c)) = \text{idifference}(\text{iplus}(a, b), c)$

THEOREM: idifference-x-x
$\text{idifference}(x, x) = \mathtt{0}$

THEOREM: idifference-iplus-canonicalizer1
$\text{idifference}(\text{iplus}(a, b), c) = \text{iplus}(a, \text{idifference}(b, c))$

THEOREM: idifference-iplus
$(\text{idifference}(\text{iplus}(x, y), x) = \text{fix-int}(y))$
$\land \quad (\text{idifference}(\text{iplus}(y, x), x) = \text{fix-int}(y))$

THEOREM: idifference-right-id
$(\neg\, \text{integerp}(b)) \rightarrow (\text{idifference}(a, b) = \text{fix-int}(a))$

THEOREM: idifference-idifference
$\text{idifference}(\text{idifference}(x, y), z) = \text{idifference}(x, \text{iplus}(y, z))$

THEOREM: idifference-fix-int
$\text{idifference}(a, \text{fix-int}(b)) = \text{idifference}(a, b)$

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

THEOREM: idifference-cancellation-1
$(\text{idifference}(x, y) = \text{idifference}(z, y)) = (\text{fix-int}(x) = \text{fix-int}(z))$

THEOREM: equal-idifference-0
$(\mathtt{0} = \text{idifference}(x, y)) = (\text{fix-int}(x) = \text{fix-int}(y))$

DEFINITION:
iplus-fringe (x)
= **if** listp $(x) \land (\text{car}(x) = \mathtt{'iplus})$
then append (iplus-fringe (cadr (x)), iplus-fringe (caddr (x)))
else list (x) **endif**

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

DEFINITION:
cancel-iplus(x)

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

THEOREM: integerp-eval\$-iplus
$(\mathrm{listp}(x) \land (\mathrm{car}(x) = \texttt{'iplus})) \rightarrow \mathrm{integerp}(\mathrm{eval\$}(\mathbf{t}, x, a))$

THEOREM: integerp-eval\$-iplus-tree
$\mathrm{integerp}(\mathrm{eval\$}(\mathbf{t}, \mathrm{iplus\text{-}tree}(l), a))$

THEOREM: integerp-eval\$-bridge
$(\mathrm{eval\$}(\mathbf{t}, z, a) = \mathrm{eval\$}(\mathbf{t}, \mathrm{iplus\text{-}tree}(x), a)) \rightarrow \mathrm{integerp}(\mathrm{eval\$}(\mathbf{t}, z, a))$

DEFINITION:
eval$-iplus-tree-append-induction $(a,\ b,\ c)$
$=$ **if** listp (a) **then** eval$-iplus-tree-append-induction (cdr $(a),\ b,\ c)$
else **nil** **endif**

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

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

THEOREM: iplus-tree-delete
eval$ (**t**, iplus-tree (delete $(x,\ y)),\ a)$
$=$ **if** $x \in y$ **then** idifference (eval$ (**t**, iplus-tree $(y),\ a)$, eval$ (**t**, $x,\ a)$)
else eval$ (**t**, iplus-tree $(y),\ a)$ **endif**

THEOREM: iplus-tree-bagdiff
subbagp $(x,\ y)$
$\rightarrow$ (eval$ (**t**, iplus-tree (bagdiff $(y,\ x)),\ a)$
$=$ idifference (eval$ (**t**, iplus-tree $(y),\ a)$, eval$ (**t**, iplus-tree $(x),\ a)$))

THEOREM: not-integerp-implies-not-equal-iplus
$(\neg$ integerp $(a)) \rightarrow ((a =$ iplus $(b,\ c)) =$ **f**)

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

THEOREM: iplus-idifference-idifference
iplus (idifference $(a,\ b)$, idifference $(c,\ d)$)
$=$ idifference (iplus $(a,\ c)$, iplus $(b,\ d)$)

DEFINITION:
cancel-idifference (x)
$=$ **if** listp $(x) \land$ (car $(x) =$ `'idifference`)
then **if** listp (cadr (x))
$\land$ (caadr $(x) =$ `'iplus`)
$\land$ listp (caddr (x))
$\land$ (caaddr $(x) =$ `'iplus`)
then list (`'idifference`,
iplus-tree (bagdiff (iplus-fringe (cadr (x)),
bagint (iplus-fringe (cadr (x)),
iplus-fringe (caddr (x))))),
iplus-tree (bagdiff (iplus-fringe (caddr (x)),
bagint (iplus-fringe (cadr (x)),

$\text{iplus-fringe}\,(\text{caddr}\,(x))))))$

elseif $\text{listp}\,(\text{cadr}\,(x))$
$\quad\land\quad (\text{caadr}\,(x) = \texttt{'iplus})$
$\quad\land\quad (\text{caddr}\,(x) \in \text{iplus-fringe}\,(\text{cadr}\,(x)))$
then $\text{iplus-tree}\,(\text{delete}\,(\text{caddr}\,(x),\ \text{iplus-fringe}\,(\text{cadr}\,(x))))$
elseif $\text{listp}\,(\text{caddr}\,(x))$
$\quad\land\quad (\text{caaddr}\,(x) = \texttt{'iplus})$
$\quad\land\quad (\text{cadr}\,(x) \in \text{iplus-fringe}\,(\text{caddr}\,(x)))$
then $\text{list}\,(\texttt{'ineg},$
$\qquad \text{iplus-tree}\,(\text{delete}\,(\text{cadr}\,(x),\ \text{iplus-fringe}\,(\text{caddr}\,(x)))))$
else x **endif**
else x **endif**

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

THEOREM: itimes-zero1
$\text{izerop}\,(x) \rightarrow (\text{itimes}\,(x,\ y) = \texttt{0})$

THEOREM: itimes-zero2
$\text{izerop}\,(y) \rightarrow (\text{itimes}\,(x,\ y) = \texttt{0})$

THEOREM: itimes-fix-int1
$\text{itimes}\,(\text{fix-int}\,(a),\ b) = \text{itimes}\,(a,\ b)$

THEOREM: itimes-fix-int2
$\text{itimes}\,(a,\ \text{fix-int}\,(b)) = \text{itimes}\,(a,\ b)$

THEOREM: commutativity-of-itimes
$\text{itimes}\,(x,\ y) = \text{itimes}\,(y,\ x)$

THEOREM: itimes-distributes-over-iplus-proof
$\text{itimes}\,(x,\ \text{iplus}\,(y,\ z)) = \text{iplus}\,(\text{itimes}\,(x,\ y),\ \text{itimes}\,(x,\ z))$

THEOREM: itimes-distributes-over-iplus
$(\text{itimes}\,(x,\ \text{iplus}\,(y,\ z)) = \text{iplus}\,(\text{itimes}\,(x,\ y),\ \text{itimes}\,(x,\ z)))$
$\land\quad (\text{itimes}\,(\text{iplus}\,(x,\ y),\ z) = \text{iplus}\,(\text{itimes}\,(x,\ z),\ \text{itimes}\,(y,\ z)))$

THEOREM: commutativity2-of-itimes
$\text{itimes}\,(x,\ \text{itimes}\,(y,\ z)) = \text{itimes}\,(y,\ \text{itimes}\,(x,\ z))$

THEOREM: associativity-of-itimes
$\text{itimes}\,(\text{itimes}\,(x,\ y),\ z) = \text{itimes}\,(x,\ \text{itimes}\,(y,\ z))$

THEOREM: itimes-distributes-over-idifference-proof
$\text{itimes}\,(\text{idifference}\,(a,\ b),\ c) = \text{idifference}\,(\text{itimes}\,(a,\ c),\ \text{itimes}\,(b,\ c))$

THEOREM: itimes-distributes-over-idifference

$$(\text{itimes}\,(\text{idifference}\,(a,\ b),\ c) = \text{idifference}\,(\text{itimes}\,(a,\ c),\ \text{itimes}\,(b,\ c)))$$
$$\land \quad (\text{itimes}\,(a,\ \text{idifference}\,(b,\ c)) = \text{idifference}\,(\text{itimes}\,(a,\ b),\ \text{itimes}\,(a,\ c)))$$

THEOREM: equal-itimes-0

$$(\text{itimes}\,(x,\ y) = \texttt{0}) = (\text{izerop}\,(x) \lor \text{izerop}\,(y))$$

THEOREM: equal-itimes-1

$$(\text{itimes}\,(a,\ b) = \texttt{1})$$
$$= \quad (((a = \texttt{1}) \land (b = \texttt{1})) \lor ((a = \texttt{-1}) \land (b = \texttt{-1})))$$

THEOREM: equal-itimes-minus-1

$$(\text{itimes}\,(a,\ b) = \texttt{-1})$$
$$= \quad (((a = \texttt{-1}) \land (b = \texttt{1})) \lor ((a = \texttt{1}) \land (b = \texttt{-1})))$$

THEOREM: itimes-1-arg1

$$\text{itimes}\,(\texttt{1},\ x) = \text{fix-int}\,(x)$$

THEOREM: quotient-remainder-uniqueness

$$((a = (r + (b * q))) \land (r < b))$$
$$\rightarrow \quad ((\text{fix}\,(r) = (a \ \mathbf{mod}\ b)) \land (\text{fix}\,(q) = (a \div b)))$$

DEFINITION:

iquotient $(i,\ j)$

$=$ **if** izerop (j) **then** `0`
  elseif negativep (i)
  then if negativep (j)
    then if (negative-guts (i) **mod** negative-guts (j)) $=$ `0`
      then negative-guts $(i) \div$ negative-guts (j)
      else $1 +$ (negative-guts $(i) \div$ negative-guts (j)) **endif**
    elseif (negative-guts (i) **mod** j) $=$ `0`
    then fix-int $(-$ (negative-guts $(i) \div j$))
    else fix-int $(-$ $(1 +$ (negative-guts $(i) \div j$))) **endif**
  elseif negativep (j) **then** fix-int $(-$ $(i \div$ negative-guts (j)))
  else $i \div j$ **endif**

DEFINITION:

$$\text{iremainder}\,(i,\ j) = \text{idifference}\,(\text{fix-int}\,(i),\ \text{itimes}\,(j,\ \text{iquotient}\,(i,\ j)))$$

THEOREM: division-theorem-part1

$$\text{integerp}\,(i) \rightarrow (\text{iplus}\,(\text{iremainder}\,(i,\ j),\ \text{itimes}\,(j,\ \text{iquotient}\,(i,\ j))) = i)$$

THEOREM: division-theorem-part2

$$(\text{integerp}\,(j) \land (j \neq \texttt{0})) \rightarrow \text{ileq}\,(\texttt{0},\ \text{iremainder}\,(i,\ j))$$

THEOREM: division-theorem-part3

$$(\text{integerp}\,(j) \land (j \neq \texttt{0})) \rightarrow \text{ilessp}\,(\text{iremainder}\,(i,\ j),\ \text{iabs}\,(j))$$

THEOREM: division-theorem

$$
\begin{array}{l}
(\text{integerp}(i) \land \text{integerp}(j) \land (j \neq 0)) \\
\rightarrow \quad ((\text{iplus}(\text{iremainder}(i, j), \text{itimes}(j, \text{iquotient}(i, j))) = i) \\
\qquad \land \quad \text{ileq}(0, \text{iremainder}(i, j)) \\
\qquad \land \quad \text{ilessp}(\text{iremainder}(i, j), \text{iabs}(j)))
\end{array}
$$

THEOREM: quotient-difference-lessp-arg2

$$
\begin{array}{l}
(((a \ \mathbf{mod}\ c) = 0) \land (b < c)) \\
\rightarrow \quad (((a - b) \div c) \\
\qquad = \quad \mathbf{if}\ b \simeq 0\ \ \mathbf{then}\ a \div c \\
\qquad\qquad \mathbf{elseif}\ b < a\ \ \mathbf{then}\ (a \div c) - (1 + (b \div c)) \\
\qquad\qquad \mathbf{else}\ 0\ \mathbf{endif})
\end{array}
$$

THEOREM: iquotient-iremainder-uniqueness

$$
\begin{array}{l}
(\text{integerp}(i) \\
\land \quad \text{integerp}(j) \\
\land \quad \text{integerp}(r) \\
\land \quad \text{integerp}(q) \\
\land \quad (j \neq 0) \\
\land \quad (i = \text{iplus}(r, \text{itimes}(j, q))) \\
\land \quad \text{ileq}(0, r) \\
\land \quad \text{ilessp}(r, \text{iabs}(j))) \\
\rightarrow \quad ((r = \text{iremainder}(i, j)) \land (q = \text{iquotient}(i, j)))
\end{array}
$$

DEFINITION:

$$
\begin{array}{l}
\text{idiv}(i, j) \\
= \quad \mathbf{if}\ \text{izerop}(j)\ \ \mathbf{then}\ 0 \\
\qquad \mathbf{elseif}\ \text{negativep}(i) \\
\qquad \mathbf{then}\ \mathbf{if}\ \text{negativep}(j)\ \ \mathbf{then}\ \text{negative-guts}(i) \div \text{negative-guts}(j) \\
\qquad\qquad \mathbf{elseif}\ (\text{negative-guts}(i)\ \mathbf{mod}\ j) = 0 \\
\qquad\qquad \mathbf{then}\ \text{fix-int}(-\ (\text{negative-guts}(i) \div j)) \\
\qquad\qquad \mathbf{else}\ \text{fix-int}(-\ (1 + (\text{negative-guts}(i) \div j)))\ \mathbf{endif} \\
\qquad \mathbf{elseif}\ \text{negativep}(j) \\
\qquad \mathbf{then}\ \mathbf{if}\ (i\ \mathbf{mod}\ \text{negative-guts}(j)) = 0 \\
\qquad\qquad \mathbf{then}\ \text{fix-int}(-\ (i \div \text{negative-guts}(j))) \\
\qquad\qquad \mathbf{else}\ \text{fix-int}(-\ (1 + (i \div \text{negative-guts}(j))))\ \mathbf{endif} \\
\qquad \mathbf{else}\ i \div j\ \mathbf{endif}
\end{array}
$$

DEFINITION:

$$\text{imod}(i, j) = \text{idifference}(\text{fix-int}(i), \text{itimes}(j, \text{idiv}(i, j)))$$

THEOREM: division-theorem-for-truncate-to-neginf-part1

$$\text{integerp}(i) \rightarrow (\text{iplus}(\text{imod}(i, j), \text{itimes}(j, \text{idiv}(i, j))) = i)$$

THEOREM: division-theorem-for-truncate-to-neginf-part2

$$\text{ilessp}(0, j) \rightarrow (\text{ileq}(0, \text{imod}(i, j)) \land \text{ilessp}(\text{imod}(i, j), j))$$

THEOREM: division-theorem-for-truncate-to-neginf-part3
(integerp (j) ∧ ilessp $(j,$ 0))
→ (ileq (imod (i, j), 0) ∧ ilessp $(j,$ imod (i, j)))

THEOREM: division-theorem-for-truncate-to-neginf
(integerp (i) ∧ integerp (j) ∧ $(j \neq$ 0))
→ ((iplus (imod (i, j), itimes $(j,$ idiv (i, j))) $= i$)
∧ **if** ilessp (0, j)
then ileq (0, imod (i, j)) ∧ ilessp (imod (i, j), j)
else ileq (imod (i, j), 0) ∧ ilessp $(j,$ imod (i, j)) **endif**)

THEOREM: idiv-imod-uniqueness
(integerp (i)
∧ integerp (j)
∧ integerp (r)
∧ integerp (q)
∧ $(j \neq$ 0)
∧ $(i =$ iplus $(r,$ itimes (j, q)))
∧ **if** ilessp (0, j) **then** ileq (0, r) ∧ ilessp (r, j)
else ileq $(r,$ 0) ∧ ilessp (j, r) **endif**)
→ (($r =$ imod (i, j)) ∧ ($q =$ idiv (i, j)))

DEFINITION:
iquo (i, j)
= **if** izerop (j) **then** 0
elseif negativep (i)
then if negativep (j) **then** negative-guts $(i) \div$ negative-guts (j)
else fix-int (− (negative-guts $(i) \div j$)) **endif**
elseif negativep (j) **then** fix-int (− ($i \div$ negative-guts (j)))
else $i \div j$ **endif**

DEFINITION:
irem (i, j) = idifference (fix-int (i), itimes $(j,$ iquo (i, j)))

THEOREM: division-theorem-for-truncate-to-zero-part1
integerp (i) → (iplus (irem (i, j), itimes $(j,$ iquo (i, j))) $= i$)

THEOREM: division-theorem-for-truncate-to-zero-part2
(integerp (i) ∧ integerp (j) ∧ $(j \neq$ 0) ∧ ileq (0, i))
→ (ileq (0, irem (i, j)) ∧ ilessp (irem (i, j), iabs (j)))

THEOREM: division-theorem-for-truncate-to-zero-part3
(integerp (i) ∧ integerp (j) ∧ $(j \neq$ 0) ∧ ilessp $(i,$ 0))
→ (ileq (irem (i, j), 0) ∧ ilessp (ineg (iabs (j)), irem (i, j)))

THEOREM: division-theorem-for-truncate-to-zero
$(\text{integerp}(i) \land \text{integerp}(j) \land (j \neq 0))$
$\rightarrow \quad ((\text{iplus}(\text{irem}(i, j), \text{itimes}(j, \text{iquo}(i, j))) = i)$
$\quad \land \quad \textbf{if } \text{ileq}(0, i)$
$\quad\quad \textbf{then } \text{ileq}(0, \text{irem}(i, j)) \land \text{ilessp}(\text{irem}(i, j), \text{iabs}(j))$
$\quad\quad \textbf{else } \text{ileq}(\text{irem}(i, j), 0)$
$\quad\quad\quad \land \quad \text{ilessp}(\text{ineg}(\text{iabs}(j)), \text{irem}(i, j)) \textbf{ endif})$

THEOREM: iquo-irem-uniqueness
$(\text{integerp}(i)$
$\land \quad \text{integerp}(j)$
$\land \quad \text{integerp}(r)$
$\land \quad \text{integerp}(q)$
$\land \quad (j \neq 0)$
$\land \quad (i = \text{iplus}(r, \text{itimes}(j, q)))$
$\land \quad \textbf{if } \text{ileq}(0, i) \textbf{ then } \text{ileq}(0, r) \land \text{ilessp}(r, \text{iabs}(j))$
$\quad\quad \textbf{else } \text{ileq}(r, 0) \land \text{ilessp}(\text{ineg}(\text{iabs}(j)), r) \textbf{ endif})$
$\rightarrow \quad ((r = \text{irem}(i, j)) \land (q = \text{iquo}(i, j)))$

THEOREM: integerp-iquotient
$\text{integerp}(\text{iquotient}(i, j))$

THEOREM: integerp-iremainder
$\text{integerp}(\text{iremainder}(i, j))$

THEOREM: integerp-idiv
$\text{integerp}(\text{idiv}(i, j))$

THEOREM: integerp-imod
$\text{integerp}(\text{imod}(i, j))$

THEOREM: integerp-iquo
$\text{integerp}(\text{iquo}(i, j))$

THEOREM: integerp-irem
$\text{integerp}(\text{irem}(i, j))$

THEOREM: iquotient-fix-int1
$\text{iquotient}(\text{fix-int}(i), j) = \text{iquotient}(i, j)$

THEOREM: iquotient-fix-int2
$\text{iquotient}(i, \text{fix-int}(j)) = \text{iquotient}(i, j)$

THEOREM: iremainder-fix-int1
$\text{iremainder}(\text{fix-int}(i), j) = \text{iremainder}(i, j)$

THEOREM: iremainder-fix-int2
iremainder $(i,\ \text{fix-int}\,(j)) = \text{iremainder}\,(i,\ j)$

THEOREM: idiv-fix-int1
idiv $(\text{fix-int}\,(i),\ j) = \text{idiv}\,(i,\ j)$

THEOREM: idiv-fix-int2
idiv $(i,\ \text{fix-int}\,(j)) = \text{idiv}\,(i,\ j)$

THEOREM: imod-fix-int1
imod $(\text{fix-int}\,(i),\ j) = \text{imod}\,(i,\ j)$

THEOREM: imod-fix-int2
imod $(i,\ \text{fix-int}\,(j)) = \text{imod}\,(i,\ j)$

THEOREM: iquo-fix-int1
iquo $(\text{fix-int}\,(i),\ j) = \text{iquo}\,(i,\ j)$

THEOREM: iquo-fix-int2
iquo $(i,\ \text{fix-int}\,(j)) = \text{iquo}\,(i,\ j)$

THEOREM: irem-fix-int1
irem $(\text{fix-int}\,(i),\ j) = \text{irem}\,(i,\ j)$

THEOREM: irem-fix-int2
irem $(i,\ \text{fix-int}\,(j)) = \text{irem}\,(i,\ j)$

THEOREM: fix-int-iquotient
fix-int $(\text{iquotient}\,(i,\ j)) = \text{iquotient}\,(i,\ j)$

THEOREM: fix-int-iremainder
fix-int $(\text{iremainder}\,(i,\ j)) = \text{iremainder}\,(i,\ j)$

THEOREM: fix-int-idiv
fix-int $(\text{idiv}\,(i,\ j)) = \text{idiv}\,(i,\ j)$

THEOREM: fix-int-imod
fix-int $(\text{imod}\,(i,\ j)) = \text{imod}\,(i,\ j)$

THEOREM: fix-int-iquo
fix-int $(\text{iquo}\,(i,\ j)) = \text{iquo}\,(i,\ j)$

THEOREM: fix-int-irem
fix-int $(\text{irem}\,(i,\ j)) = \text{irem}\,(i,\ j)$

EVENT: Enable irem; name this event 'irem-off'.

EVENT: Enable iquo; name this event 'iquo-off'.

EVENT: Enable imod; name this event 'imod-off'.

EVENT: Enable idiv; name this event 'idiv-off'.

EVENT: Enable quotient-difference-lessp-arg2; name this event 'quotient-difference-lessp-arg2-off'.

EVENT: Enable iremainder; name this event 'iremainder-off'.

EVENT: Enable iquotient; name this event 'iquotient-off'.

EVENT: Enable cancel-idifference; name this event 'cancel-idifference-off'.

EVENT: Enable not-integerp-implies-not-equal-iplus; name this event 'not-integerp-implies-not-equal-iplus-off'.

EVENT: Enable iplus-tree-bagdiff; name this event 'iplus-tree-bagdiff-off'.

EVENT: Enable iplus-tree-delete; name this event 'iplus-tree-delete-off'.

EVENT: Enable iplus-tree-iplus-fringe; name this event 'iplus-tree-iplus-fringe-off'.

EVENT: Enable eval$-iplus-tree-append; name this event 'eval$-iplus-tree-append-off'.

EVENT: Enable integerp-eval$-bridge; name this event 'integerp-eval$-bridge-off'.

EVENT: Enable integerp-eval$-iplus-tree; name this event 'integerp-eval$-iplus-tree-off'.

EVENT: Enable integerp-eval$-iplus; name this event 'integerp-eval$-iplus-off'.

EVENT: Enable cancel-iplus; name this event 'cancel-iplus-off'.

EVENT: Enable itimes; name this event 'itimes-off'.

EVENT: Enable iabs; name this event 'iabs-off'.

EVENT: Enable idifference; name this event 'idifference-off'.

EVENT: Enable ineg; name this event 'ineg-off'.

EVENT: Enable iplus; name this event 'iplus-off'.

EVENT: Enable ileq; name this event 'ileq-off'.

EVENT: Enable ilessp; name this event 'ilessp-off'.

EVENT: Enable izerop; name this event 'izerop-off'.

EVENT: Enable fix-int; name this event 'fix-int-off'.

EVENT: Enable integerp; name this event 'integerp-off'.

EVENT: Let us define the theory *integers* to consist of the following events: fix-int-irem, fix-int-iquo, fix-int-imod, fix-int-idiv, fix-int-iremainder, fix-int-iquotient, irem-fix-int2, irem-fix-int1, iquo-fix-int2, iquo-fix-int1, imod-fix-int2, imod-fix-int1, idiv-fix-int2, idiv-fix-int1, iremainder-fix-int2, iremainder-fix-int1, iquotient-fix-int2, iquotient-fix-int1, integerp-irem, integerp-iquo, integerp-imod, integerp-idiv, integerp-iremainder, integerp-iquotient, itimes-1-arg1, equal-itimes-minus-1, equal-itimes-1, equal-itimes-0, itimes-distributes-over-idifference, itimes-distributes-over-idifference-proof, associativity-of-itimes, commutativity2-of-itimes, itimes-distributes-over-iplus, commutativity-of-itimes, itimes-distributes-over-iplus-proof, itimes-fix-int2, itimes-fix-int1, itimes-zero2, itimes-zero1, correctness-of-cancel-idifference, iplus-idifference-idifference, correctness-of-cancel-iplus, equal-idifference-0, idifference-cancellation-1, idifference-0, idifference-fix-int, idifference-idifference, idifference-right-id, idifference-iplus, idifference-iplus-canonicalizer1, idifference-x-x, iplus-idifference-arg2, iplus-idifference-arg1, iplus-fix-int2, iplus-fix-int1, iplus-ineg2, iplus-ineg1, iplus-cancellation, associativity-of-iplus, commutativity-of-iplus, commutativity2-of-iplus, iplus-0, iplus-right-id, iplus-left-id, ineg-fix-int, ineg-idifference, ineg-ineg, ineg-iplus, ineg-id, fix-int-itimes, fix-int-iabs, fix-int-ineg, fix-int-idifference, fix-int-iplus, fix-int-fix-int, integerp-itimes, integerp-iabs, integerp-ineg, integerp-idifference, integerp-iplus, integerp-fix-int.

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 $(x) \land$ (car (x) = `'times`)
then append (times-fringe (cadr (x)), times-fringe (caddr (x)))
else list (x) **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) $\in \mathbf{N}$)

THEOREM: eval\$-times
eval\$ ($\mathbf{t}$, list (`'times`, x, y), a) = (eval\$ ($\mathbf{t}$, x, a) $*$ eval\$ ($\mathbf{t}$, y, a))

THEOREM: eval\$-times2
(car (x) = `'times`)
$\rightarrow$ (eval\$ ($\mathbf{t}$, x, a) = (eval\$ ($\mathbf{t}$, cadr (x), a) $*$ eval\$ ($\mathbf{t}$, caddr (x), a)))

THEOREM: eval\$-or
$\text{eval\$}(\mathbf{t}, \text{list}(\texttt{'or}, x, y), a) = (\text{eval\$}(\mathbf{t}, x, a) \vee \text{eval\$}(\mathbf{t}, y, a))$

THEOREM: eval\$-or2
$(\text{car}(x) = \texttt{'or})$
$\rightarrow \quad (\text{eval\$}(\mathbf{t}, x, a) = (\text{eval\$}(\mathbf{t}, \text{cadr}(x), a) \vee \text{eval\$}(\mathbf{t}, \text{caddr}(x), a)))$

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

THEOREM: eval\$-lessp
$\text{eval\$}(\mathbf{t}, \text{list}(\texttt{'lessp}, x, y), a) = (\text{eval\$}(\mathbf{t}, x, a) < \text{eval\$}(\mathbf{t}, y, a))$

THEOREM: eval\$-lessp2
$(\text{car}(x) = \texttt{'lessp})$
$\rightarrow \quad (\text{eval\$}(\mathbf{t}, x, a) = (\text{eval\$}(\mathbf{t}, \text{cadr}(x), a) < \text{eval\$}(\mathbf{t}, \text{caddr}(x), a)))$

THEOREM: eval\$-quotient
$\text{eval\$}(\mathbf{t}, \text{list}(\texttt{'quotient}, x, y), a) = (\text{eval\$}(\mathbf{t}, x, a) \div \text{eval\$}(\mathbf{t}, y, a))$

THEOREM: eval\$-quotient2
$(\text{car}(x) = \texttt{'quotient})$
$\rightarrow \quad (\text{eval\$}(\mathbf{t}, x, a) = (\text{eval\$}(\mathbf{t}, \text{cadr}(x), a) \div \text{eval\$}(\mathbf{t}, \text{caddr}(x), a)))$

THEOREM: eval\$-if
$\text{eval\$}(\mathbf{t}, \text{list}(\texttt{'if}, x, y, z), a)$
$= \quad \textbf{if } \text{eval\$}(\mathbf{t}, x, a) \textbf{ then } \text{eval\$}(\mathbf{t}, y, a)$
$\quad\quad \textbf{else } \text{eval\$}(\mathbf{t}, z, a) \textbf{ endif}$

THEOREM: eval\$-if2
$(\text{car}(x) = \texttt{'if})$
$\rightarrow \quad (\text{eval\$}(\mathbf{t}, x, a)$
$\quad\quad = \quad \textbf{if } \text{eval\$}(\mathbf{t}, \text{cadr}(x), a) \textbf{ then } \text{eval\$}(\mathbf{t}, \text{caddr}(x), a)$
$\quad\quad\quad\quad \textbf{else } \text{eval\$}(\mathbf{t}, \text{cadddr}(x), a) \textbf{ endif})$

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

THEOREM: bagdiff-delete
$\text{bagdiff}(\text{delete}(e, x), y) = \text{delete}(e, \text{bagdiff}(x, y))$

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

THEOREM: stupid-hack
$((a \in \mathbf{N}) \wedge (b \in \mathbf{N})) \rightarrow (((a \not< b) \wedge (b \not< a)) = (a = b))$

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

Theorem: equal-times-bridge
$((a * b) = (c * a * d)) = ((a \simeq \mathtt{0}) \vee (\mathrm{fix}\,(b) = (c * d)))$

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

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

Theorem: or-zerop-tree-is-not-zerop-tree
$\mathrm{eval\$}\,(\mathbf{t},\ \text{or-zerop-tree}\,(x),\ a) = (\neg\ \mathrm{eval\$}\,(\mathbf{t},\ \text{and-not-zerop-tree}\,(x),\ a))$

Theorem: zerop-makes-times-tree-zero2
$(\mathrm{eval\$}\,(\mathbf{t},\ \text{or-zerop-tree}\,(x),\ a) \wedge \mathrm{subbagp}\,(x,\ y))$
$\rightarrow \quad (\mathrm{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(y),\ a) = \mathtt{0})$

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

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

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

THEOREM: eval$-lessp-times-tree-bagdiff

$$
\begin{array}{l}
(\text{subbagp}\,(x,\ y) \land \text{subbagp}\,(x,\ z) \land \text{eval\$}\,(\mathbf{t},\ \text{and-not-zerop-tree}\,(x),\ a))\\
\to\quad ((\text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(\text{bagdiff}\,(y,\ x)),\ a)\\
\qquad\quad < \quad \text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(\text{bagdiff}\,(z,\ x)),\ a))\\
\qquad = \quad (\text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(y),\ a) < \text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(z),\ a)))
\end{array}
$$

THEOREM: zerop-makes-lessp-false-bridge

$$
\begin{array}{l}
((\text{car}\,(x) = \texttt{'times})\\
\ \land\quad (\text{car}\,(y) = \texttt{'times})\\
\ \land\quad (\neg\ \text{eval\$}\,(\mathbf{t},\\
\qquad\qquad\quad \text{and-not-zerop-tree}\,(\text{bagint}\,(\text{times-fringe}\,(x),\ \text{times-fringe}\,(y))),\\
\qquad\qquad\quad a)))\\
\to\quad (((\text{eval\$}\,(\mathbf{t},\ \text{cadr}\,(x),\ a) * \text{eval\$}\,(\mathbf{t},\ \text{caddr}\,(x),\ a))\\
\qquad\quad < \quad (\text{eval\$}\,(\mathbf{t},\ \text{cadr}\,(y),\ a) * \text{eval\$}\,(\mathbf{t},\ \text{caddr}\,(y),\ a)))\\
\qquad = \quad \mathbf{f})
\end{array}
$$

THEOREM: correctness-of-cancel-lessp-times

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

DEFINITION:

$$
\begin{array}{l}
\text{cancel-quotient-times}\,(x)\\
= \quad \mathbf{if}\ (\text{car}\,(x) = \texttt{'quotient})\\
\qquad\quad \land\quad (\text{caadr}\,(x) = \texttt{'times})\\
\qquad\quad \land\quad (\text{caaddr}\,(x) = \texttt{'times})\\
\quad\ \mathbf{then\ if}\ \text{listp}\,(\text{bagint}\,(\text{times-fringe}\,(\text{cadr}\,(x)),\ \text{times-fringe}\,(\text{caddr}\,(x))))\\
\qquad\quad \mathbf{then}\ \text{cons}\,(\texttt{'if},\\
\qquad\qquad\qquad \text{cons}\,(\text{and-not-zerop-tree}\,(\text{bagint}\,(\text{times-fringe}\,(\text{cadr}\,(x)),\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{times-fringe}\,(\text{caddr}\,(x)))),\\
\qquad\qquad\qquad\qquad \text{cons}\,(\text{list}\,(\texttt{'quotient},\\
\qquad\qquad\qquad\qquad\qquad \text{times-tree}\,(\text{bagdiff}\,(\text{times-fringe}\,(\text{cadr}\,(x)),\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{bagint}\,(\text{times-fringe}\,(\text{cadr}\,(x)),\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{times-fringe}\,(\text{caddr}\,(x))))),\\
\qquad\qquad\qquad\qquad\qquad \text{times-tree}\,(\text{bagdiff}\,(\text{times-fringe}\,(\text{caddr}\,(x)),\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{bagint}\,(\text{times-fringe}\,(\text{cadr}\,(x)),\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{times-fringe}\,(\text{caddr}\,(x)))))),\\
\qquad\qquad\qquad\qquad\quad \texttt{'((zero))))})\\
\qquad\quad \mathbf{else}\ x\ \mathbf{endif}\\
\quad\ \mathbf{else}\ x\ \mathbf{endif}
\end{array}
$$

THEOREM: zerop-makes-quotient-zero-bridge

$$
\begin{array}{l}
((\text{car}\,(x) = \texttt{'times})\\
\ \land\quad (\text{car}\,(y) = \texttt{'times})\\
\ \land\quad (\neg\ \text{eval\$}\,(\mathbf{t},\\
\qquad\qquad\quad \text{and-not-zerop-tree}\,(\text{bagint}\,(\text{times-fringe}\,(x),\ \text{times-fringe}\,(y))),\\
\qquad\qquad\quad a)))
\end{array}
$$

$\rightarrow \quad (((\text{eval\$}(\mathbf{t}, \text{cadr}(x), a) * \text{eval\$}(\mathbf{t}, \text{caddr}(x), a))$
$\quad\quad \div \quad (\text{eval\$}(\mathbf{t}, \text{cadr}(y), a) * \text{eval\$}(\mathbf{t}, \text{caddr}(y), a)))$
$\quad = \quad \mathtt{0})$

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

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

DEFINITION:
cancel-equal-times (x)
$= \quad \mathbf{if}\ (\text{car}(x) = \mathtt{'equal})$
$\quad\quad \land \quad (\text{caadr}(x) = \mathtt{'times})$
$\quad\quad \land \quad (\text{caaddr}(x) = \mathtt{'times})$
$\quad \mathbf{then\ if}\ \text{listp}(\text{bagint}(\text{times-fringe}(\text{cadr}(x)), \text{times-fringe}(\text{caddr}(x))))$
$\quad\quad \mathbf{then}\ \text{list}(\mathtt{'or},$
$\quad\quad\quad\quad \text{or-zerop-tree}(\text{bagint}(\text{times-fringe}(\text{cadr}(x)),$
$\quad\quad\quad\quad\quad\quad \text{times-fringe}(\text{caddr}(x)))),$
$\quad\quad\quad\quad \text{list}(\mathtt{'equal},$
$\quad\quad\quad\quad\quad \text{times-tree}(\text{bagdiff}(\text{times-fringe}(\text{cadr}(x)),$
$\quad\quad\quad\quad\quad\quad\quad \text{bagint}(\text{times-fringe}(\text{cadr}(x)),$
$\quad\quad\quad\quad\quad\quad\quad\quad \text{times-fringe}(\text{caddr}(x))))),$
$\quad\quad\quad\quad\quad \text{times-tree}(\text{bagdiff}(\text{times-fringe}(\text{caddr}(x)),$
$\quad\quad\quad\quad\quad\quad\quad \text{bagint}(\text{times-fringe}(\text{cadr}(x)),$
$\quad\quad\quad\quad\quad\quad\quad\quad \text{times-fringe}(\text{caddr}(x)))))))$
$\quad\quad \mathbf{else}\ x\ \mathbf{endif}$
$\quad \mathbf{else}\ x\ \mathbf{endif}$

THEOREM: zerop-makes-equal-true-bridge
$((\text{car}(x) = \mathtt{'times})$
$\ \land \quad (\text{car}(y) = \mathtt{'times})$
$\ \land \quad \text{eval\$}(\mathbf{t}, \text{or-zerop-tree}(\text{bagint}(\text{times-fringe}(x), \text{times-fringe}(y))), a))$
$\rightarrow \quad (((\text{eval\$}(\mathbf{t}, \text{cadr}(x), a) * \text{eval\$}(\mathbf{t}, \text{caddr}(x), a))$
$\quad\quad = \quad (\text{eval\$}(\mathbf{t}, \text{cadr}(y), a) * \text{eval\$}(\mathbf{t}, \text{caddr}(y), a)))$
$\quad = \quad \mathbf{t})$

THEOREM: eval\$-equal-times-tree-bagdiff
$(\text{subbagp}(x, y) \land \text{subbagp}(x, z) \land (\neg\ \text{eval\$}(\mathbf{t}, \text{or-zerop-tree}(x), a)))$
$\rightarrow \quad ((\text{eval\$}(\mathbf{t}, \text{times-tree}(\text{bagdiff}(y, x)), a)$
$\quad\quad = \quad \text{eval\$}(\mathbf{t}, \text{times-tree}(\text{bagdiff}(z, x)), a))$
$\quad = \quad (\text{eval\$}(\mathbf{t}, \text{times-tree}(y), a) = \text{eval\$}(\mathbf{t}, \text{times-tree}(z), a)))$

THEOREM: cancel-equal-times-preserves-inequality

$$
\begin{array}{ll}
(\text{subbagp}\,(z,\ x) & \\
\ \land & \text{subbagp}\,(z,\ y) \\
\ \land & (\text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(x),\ a) \neq \text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(y),\ a))) \\
\rightarrow & (\text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(\text{bagdiff}\,(x,\ z)),\ a) \\
 & \quad \neq \quad \text{eval\$}\,(\mathbf{t},\ \text{times-tree}\,(\text{bagdiff}\,(y,\ z)),\ a))
\end{array}
$$

THEOREM: cancel-equal-times-preserves-inequality-bridge

$$
\begin{array}{ll}
((\text{car}\,(x) = \texttt{'times}) & \\
\ \land & (\text{car}\,(y) = \texttt{'times}) \\
\ \land & ((\text{eval\$}\,(\mathbf{t},\ \text{cadr}\,(x),\ a) * \text{eval\$}\,(\mathbf{t},\ \text{caddr}\,(x),\ a)) \\
 & \quad \neq \quad (\text{eval\$}\,(\mathbf{t},\ \text{cadr}\,(y),\ a) * \text{eval\$}\,(\mathbf{t},\ \text{caddr}\,(y),\ a)))) \\
\rightarrow & (\text{eval\$}\,(\mathbf{t}, \\
 & \qquad \text{times-tree}\,(\text{bagdiff}\,(\text{times-fringe}\,(x), \\
 & \qquad\qquad \text{bagint}\,(\text{times-fringe}\,(x),\ \text{times-fringe}\,(y)))), \\
 & \qquad a) \\
 & \quad \neq \quad \text{eval\$}\,(\mathbf{t}, \\
 & \qquad\qquad \text{times-tree}\,(\text{bagdiff}\,(\text{times-fringe}\,(y), \\
 & \qquad\qquad\qquad \text{bagint}\,(\text{times-fringe}\,(x), \\
 & \qquad\qquad\qquad\qquad \text{times-fringe}\,(y)))), \\
 & \qquad\qquad a))
\end{array}
$$

THEOREM: correctness-of-cancel-equal-times

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

EVENT: Enable numberp-eval\$-times; name this event ‘numberp-eval\$-times-off’.

EVENT: Enable eval\$-times; name this event ‘eval\$-times-off’.

EVENT: Enable eval\$-times2; name this event ‘eval\$-times2-off’.

EVENT: Enable eval\$-or; name this event ‘eval\$-or-off’.

EVENT: Enable eval\$-or2; name this event ‘eval\$-or2-off’.

EVENT: Enable eval\$-equal; name this event ‘eval\$-equal-off’.

EVENT: Enable eval\$-lessp; name this event ‘eval\$-lessp-off’.

EVENT: Enable eval\$-lessp2; name this event ‘eval\$-lessp2-off’.

EVENT: Enable eval$-if; name this event 'eval$-if-off'.

EVENT: Enable eval$-if2; name this event 'eval$-if2-off'.

EVENT: Enable eval$-quotient; name this event 'eval$-quotient-off'.

EVENT: Enable eval$-quotient2; name this event 'eval$-quotient2-off'.

EVENT: Enable numberp-eval$-times-tree; name this event 'numberp-eval$-times-tree-off'.

EVENT: Enable bagdiff-delete; name this event 'bagdiff-delete-off'.

EVENT: Enable lessp-times-arg1; name this event 'lessp-times-arg1-off'.

EVENT: Enable stupid-hack; name this event 'stupid-hack-off'.

EVENT: Enable equal-times-arg1; name this event 'equal-times-arg1-off'.

EVENT: Enable equal-times-bridge; name this event 'equal-times-bridge-off'.

EVENT: Enable eval$-times-member; name this event 'eval$-times-member-off'.

EVENT: Enable zerop-makes-times-tree-zero; name this event 'zerop-makes-times-tree-zero-off'.

EVENT: Enable or-zerop-tree-is-not-zerop-tree; name this event 'or-zerop-tree-is-not-zerop-tree-off'.

EVENT: Enable zerop-makes-times-tree-zero2; name this event 'zerop-makes-times-tree-zero2-off'.

EVENT: Enable times-tree-append; name this event 'times-tree-append-off'.

EVENT: Enable times-tree-of-times-fringe; name this event 'times-tree-of-times-fringe-off'.

EVENT: Enable eval$-lessp-times-tree-bagdiff; name this event 'eval$-lessp-times-tree-bagdiff-off'.

EVENT: Enable zerop-makes-lessp-false-bridge; name this event 'zerop-makes-lessp-false-bridge-off'.

EVENT: Enable zerop-makes-quotient-zero-bridge; name this event 'zerop-makes-quotient-zero-bridge-off'.

EVENT: Enable eval$-quotient-times-tree-bagdiff; name this event 'eval$-quotient-times-tree-bagdiff-off'.

EVENT: Enable zerop-makes-equal-true-bridge; name this event 'zerop-makes-equal-true-bridge-off'.

EVENT: Enable eval$-equal-times-tree-bagdiff; name this event 'eval$-equal-times-tree-bagdiff-off'.

EVENT: Enable cancel-equal-times-preserves-inequality; name this event 'cancel-equal-times-preserves-inequality-off'.

EVENT: Enable cancel-equal-times-preserves-inequality-bridge; name this event 'cancel-equal-times-preserves-inequality-bridge-off'.

EVENT: Add the shell *rational*, with recognizer function symbol *rational-formp* and 2 accessors: *numerator*, with type restriction (one-of numberp negativep) and default value zero; *denominator*, with type restriction (one-of numberp) and default value zero.

EVENT: Let us define the theory *rational-defns* to consist of the following events: count-rational, numerator-denominator-elim, rational-numerator-denominator, rational-equal, denominator-lesseqp, denominator-lessp, denominator-type-restriction, denominator-nrational-formp, denominator-rational, numerator-lesseqp, numerator-lessp, numerator-type-restriction, numerator-nrational-formp, numerator-rational, denominator, numerator, *1*rational-formp, rational-formp.

DEFINITION:
rationalp (x)
= (rational-formp (x)

$$
\begin{array}{l}
\quad\land\quad \text{integerp}\,(\text{numerator}\,(x))\\
\quad\land\quad (\text{denominator}\,(x) \not\simeq \mathtt{0}))
\end{array}
$$

DEFINITION:
fix-rational (x)
$= \;$ **if** rationalp (x) **then** x
else rational $(\mathtt{0},\ \mathtt{1})$ **endif**

DEFINITION:
reduce (r)

$$
\begin{array}{l}
= \;\; \textbf{if}\ \text{rationalp}\,(r)\\
\quad\;\; \textbf{then if}\ \text{negativep}\,(\text{numerator}\,(r))\\
\qquad\qquad \textbf{then}\ \text{rational}\,(-\ (\text{negative-guts}\,(\text{numerator}\,(r))\\
\qquad\qquad\qquad\qquad\qquad \div\ \ \text{gcd}\,(\text{negative-guts}\,(\text{numerator}\,(r)),\\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{denominator}\,(r))),\\
\qquad\qquad\qquad\qquad\quad \text{denominator}\,(r)\\
\qquad\qquad\qquad\qquad\quad \div\ \ \text{gcd}\,(\text{negative-guts}\,(\text{numerator}\,(r)),\\
\qquad\qquad\qquad\qquad\qquad\qquad \text{denominator}\,(r)))\\
\qquad\qquad \textbf{else}\ \text{rational}\,(\text{numerator}\,(r)\\
\qquad\qquad\qquad\qquad\quad \div\ \ \text{gcd}\,(\text{numerator}\,(r),\ \text{denominator}\,(r)),\\
\qquad\qquad\qquad\qquad\quad \text{denominator}\,(r)\\
\qquad\qquad\qquad\qquad\quad \div\ \ \text{gcd}\,(\text{numerator}\,(r),\ \text{denominator}\,(r)))\ \textbf{endif}\\
\quad\;\; \textbf{else}\ \text{rational}\,(\mathtt{0},\ \mathtt{1})\ \textbf{endif}
\end{array}
$$

DEFINITION:
simple-rplus $(x,\ y)$

$$
\begin{array}{l}
= \;\; \text{rational}\,(\text{iplus}\,(\text{itimes}\,(\text{numerator}\,(\text{fix-rational}\,(x)),\\
\qquad\qquad\qquad\qquad\qquad\quad \text{denominator}\,(\text{fix-rational}\,(y))),\\
\qquad\qquad\qquad\qquad \text{itimes}\,(\text{numerator}\,(\text{fix-rational}\,(y)),\\
\qquad\qquad\qquad\qquad\qquad\quad \text{denominator}\,(\text{fix-rational}\,(x)))),\\
\qquad\qquad\quad \text{itimes}\,(\text{denominator}\,(\text{fix-rational}\,(x)),\\
\qquad\qquad\qquad\quad \text{denominator}\,(\text{fix-rational}\,(y))))
\end{array}
$$

DEFINITION: rplus $(x,\ y) =$ reduce (simple-rplus $(x,\ y)$)

DEFINITION:
simple-rneg (x)

$$
\begin{array}{l}
= \;\; \text{rational}\,(\text{ineg}\,(\text{numerator}\,(\text{fix-rational}\,(x))),\\
\qquad\qquad\quad \text{denominator}\,(\text{fix-rational}\,(x)))
\end{array}
$$

DEFINITION: rneg $(x) =$ reduce (simple-rneg (x))

DEFINITION: rdifference $(a,\ b) =$ rplus $(a,$ rneg $(b))$

DEFINITION:

simple-rtimes $(x,\ y)$
$=$ rational (itimes (numerator (fix-rational (x)), numerator (fix-rational (y))),
denominator (fix-rational (x))
$*$ denominator (fix-rational (y)))

DEFINITION: rtimes $(x,\ y)$ = reduce (simple-rtimes $(x,\ y)$)

DEFINITION:
rzerop (x) = (($\neg$ rationalp (x)) $\vee$ (numerator (x) = 0))

DEFINITION:
simple-rinverse (r)
$=$ **if** rzerop (r) **then** rational (0, 1)
elseif negativep (numerator (r))
then rational (ineg (denominator (r)), ineg (numerator (r)))
else rational (denominator (r), numerator (r)) **endif**

DEFINITION: rinverse (r) = reduce (simple-rinverse (r))

DEFINITION: rquotient $(x,\ y)$ = rtimes $(x$, rinverse (y))

DEFINITION:
simple-rmagnitude (x)
$=$ **if** negativep (numerator (fix-rational (x))) **then** rneg (fix-rational (x))
else fix-rational (x) **endif**

DEFINITION: rmagnitude (x) = reduce (simple-rmagnitude (x))

DEFINITION:
rlessp $(x,\ y)$
$=$ ilessp (itimes (numerator (fix-rational (x)), denominator (fix-rational (y))),
itimes (numerator (fix-rational (y)), denominator (fix-rational (x))))

DEFINITION:
requal $(x,\ y)$
$=$ (itimes (numerator (fix-rational (x)), denominator (fix-rational (y)))
$=$ itimes (numerator (fix-rational (y)),
denominator (fix-rational (x))))

THEOREM: integerp-minus
integerp $(-\ x)$ = $(0 < x)$

THEOREM: fix-int-on-integers
integerp (x) $\rightarrow$ (fix-int (x) = x)

THEOREM: itimes-ineg-arg1
itimes (ineg (x), y) = ineg (itimes $(x,\ y)$)

THEOREM: itimes-ineg-arg2
itimes $(x,\ \text{ineg}(y)) = \text{ineg}(\text{itimes}(x,\ y))$

THEOREM: integerp-if-negativep-non-zero
$(\text{negativep}(x) \land (x \neq (-\ \mathtt{0}))) \rightarrow \text{integerp}(x)$

THEOREM: integerp-if-numberp
$(x \in \mathbf{N}) \rightarrow \text{integerp}(x)$

THEOREM: itimes-negativep-arg1
$\text{negativep}(x) \rightarrow (\text{itimes}(x,\ y) = \text{ineg}(\text{itimes}(\text{negative-guts}(x),\ y)))$

THEOREM: itimes-negativep-arg2
$\text{negativep}(y) \rightarrow (\text{itimes}(x,\ y) = \text{ineg}(\text{itimes}(x,\ \text{negative-guts}(y))))$

THEOREM: equal-ineg-ineg
$(\text{integerp}(x) \land \text{integerp}(y)) \rightarrow ((\text{ineg}(x) = \text{ineg}(y)) = (x = y))$

THEOREM: itimes-is-times
$((x \in \mathbf{N}) \land (y \in \mathbf{N})) \rightarrow (\text{itimes}(x,\ y) = (x * y))$

THEOREM: iplus-is-plus
$((x \in \mathbf{N}) \land (y \in \mathbf{N})) \rightarrow (\text{iplus}(x,\ y) = (x + y))$

THEOREM: rationalp-reduce
$\text{rationalp}(\text{reduce}(x))$

THEOREM: rationalp-rplus
$\text{rationalp}(\text{rplus}(x,\ y))$

THEOREM: rationalp-fix-rational
$\text{rationalp}(\text{fix-rational}(x))$

THEOREM: rationalp-rtimes
$\text{rationalp}(\text{rtimes}(x,\ y))$

THEOREM: rationalp-rneg
$\text{rationalp}(\text{rneg}(x))$

THEOREM: rationalp-rdifference
$\text{rationalp}(\text{rdifference}(x,\ y))$

THEOREM: rationalp-rquotient
$\text{rationalp}(\text{rquotient}(x,\ y))$

THEOREM: rationalp-rmagnitude
$\text{rationalp}(\text{rmagnitude}(x))$

THEOREM: fix-rational-reduce
$\text{fix-rational}\,(\text{reduce}\,(x)) = \text{reduce}\,(x)$

THEOREM: fix-rational-rplus
$\text{fix-rational}\,(\text{rplus}\,(x,\ y)) = \text{rplus}\,(x,\ y)$

THEOREM: fix-rational-fix-rational
$\text{fix-rational}\,(\text{fix-rational}\,(x)) = \text{fix-rational}\,(x)$

THEOREM: fix-rational-rtimes
$\text{fix-rational}\,(\text{rtimes}\,(x,\ y)) = \text{rtimes}\,(x,\ y)$

THEOREM: fix-rational-rneg
$\text{fix-rational}\,(\text{rneg}\,(x)) = \text{rneg}\,(x)$

THEOREM: fix-rational-rdifference
$\text{fix-rational}\,(\text{rdifference}\,(x,\ y)) = \text{rdifference}\,(x,\ y)$

THEOREM: fix-rational-rquotient
$\text{fix-rational}\,(\text{rquotient}\,(x,\ y)) = \text{rquotient}\,(x,\ y)$

THEOREM: fix-rational-rmagnitude
$\text{fix-rational}\,(\text{rmagnitude}\,(x)) = \text{rmagnitude}\,(x)$

THEOREM: rational-generalization
$(\text{rationalp}\,(x) \rightarrow \text{integerp}\,(\text{numerator}\,(x)))$
$\wedge\quad (\text{rationalp}\,(x) \rightarrow (\text{denominator}\,(x) \in \mathbf{N}))$
$\wedge\quad (\text{rationalp}\,(x) \rightarrow (\text{denominator}\,(x) \not\simeq \mathtt{0}))$

THEOREM: gcd-remainder-fact1
$((\mathtt{1} < b) \wedge (\text{gcd}\,(a,\ b) = \mathtt{1})) \rightarrow ((a \ \mathbf{mod}\ b) \neq \mathtt{0})$

THEOREM: gcd-remainder-fact2
$((\mathtt{1} < b) \wedge (\text{gcd}\,(b,\ a) = \mathtt{1})) \rightarrow ((a \ \mathbf{mod}\ b) \neq \mathtt{0})$

DEFINITION:
$\text{gcd-times1-induct}\,(x,\ y)$
$=\quad \mathbf{if}\ y \simeq \mathtt{0}\ \mathbf{then}\ \mathbf{t}$
$\qquad \mathbf{else}\ \text{gcd-times1-induct}\,(x,\ y - 1)\ \mathbf{endif}$

THEOREM: gcd-times1
$\text{gcd}\,(x,\ x * y) = \text{fix}\,(x)$

THEOREM: gcd-times2
$\text{gcd}\,(y,\ x * y) = \text{fix}\,(y)$

THEOREM: gcd-quotient-quotient
$((0 < a) \wedge (0 < b)) \rightarrow (\text{gcd}\,(a \div \text{gcd}\,(a,\ b),\ b \div \text{gcd}\,(a,\ b)) = 1)$

THEOREM: divides-each-equality
$(((a \ \mathbf{mod}\ b) = 0) \wedge ((b \ \mathbf{mod}\ a) = 0)) = (\text{fix}\,(a) = \text{fix}\,(b))$

EVENT: Enable idifference-iplus-canonicalizer1; name this event 'idifference-iplus-canonicalizer1-off'.

EVENT: Disable ilessp; name this event 'ilessp-on'.

EVENT: Disable idifference; name this event 'idifference-on'.

EVENT: Disable iplus; name this event 'iplus-on'.

EVENT: Disable itimes; name this event 'itimes-on'.

EVENT: Disable ineg; name this event 'ineg-on'.

EVENT: Disable integerp; name this event 'integerp-on'.

EVENT: Disable fix-int; name this event 'fix-int-on'.

EVENT: Disable izerop; name this event 'izerop-on'.

DEFINITION:
gcd-factors $(x,\ y)$
$=$ **if** $x \simeq 0$ **then** '(0 . 1)
elseif $y \simeq 0$ **then** '(1 . 0)
elseif $x < y$
then cons (idifference (car (gcd-factors $(x,\ y - x)$),
cdr (gcd-factors $(x,\ y - x)$)),
cdr (gcd-factors $(x,\ y - x)$))
else cons (car (gcd-factors $(x - y,\ y)$),
idifference (cdr (gcd-factors $(x - y,\ y)$),
car (gcd-factors $(x - y,\ y)$))) **endif**

THEOREM: gcd-factors-gives-linear-combination
$((x \in \mathbf{N}) \wedge (y \in \mathbf{N}))$
$\rightarrow$ (iplus (itimes (car (gcd-factors $(x,\ y)$)), x),

$$\text{itimes}\,(\text{cdr}\,(\text{gcd-factors}\,(x,\ y)),\ y))$$
$$=\quad \text{gcd}\,(x,\ y))$$

Event: Disable iremainder; name this event ‘iremainder-on’.

Event: Disable iquotient; name this event ‘iquotient-on’.

Theorem: gcd-factors-gives-linear-combination-rewrite

$$((x \in \mathbf{N}) \land (y \in \mathbf{N}))$$
$$\rightarrow \quad (\text{gcd}\,(x,\ y) = \text{iplus}\,(\text{itimes}\,(\text{car}\,(\text{gcd-factors}\,(x,\ y)),\ x),\ \text{itimes}\,(\text{cdr}\,(\text{gcd-factors}\,(x,\ y)),\ y)))$$

Theorem: dpr-hack1

$$\text{itimes}\,(x,\ \text{iplus}\,(\text{itimes}\,(c,\ a),\ \text{itimes}\,(b,\ q)))$$
$$=\quad \text{iplus}\,(\text{itimes}\,(c,\ \text{itimes}\,(a,\ x)),\ \text{itimes}\,(b,\ \text{itimes}\,(q,\ x)))$$

Theorem: dpr-hack2

$$((x \in \mathbf{N}) \land (y \in \mathbf{N}) \land (c \in \mathbf{N}) \land (\text{iremainder}\,(\text{itimes}\,(c,\ y),\ x) = \mathtt{0}))$$
$$\rightarrow \quad (\text{itimes}\,(\text{iquotient}\,(\text{itimes}\,(c,\ y),\ x),\ x) = \text{itimes}\,(c,\ y))$$

Theorem: dpr-hack3

$$\text{iplus}\,(\text{itimes}\,(c,\ \text{itimes}\,(a,\ x)),\ \text{itimes}\,(b,\ \text{itimes}\,(c,\ y)))$$
$$=\quad \text{iplus}\,(\text{itimes}\,(c,\ \text{itimes}\,(a,\ x)),\ \text{itimes}\,(c,\ \text{itimes}\,(b,\ y)))$$

Theorem: dpr-hack4

$$\text{iplus}\,(\text{itimes}\,(c,\ \text{itimes}\,(a,\ x)),\ \text{itimes}\,(c,\ \text{itimes}\,(b,\ y)))$$
$$=\quad \text{itimes}\,(c,\ \text{iplus}\,(\text{itimes}\,(a,\ x),\ \text{itimes}\,(b,\ y)))$$

Theorem: dpr-hack5

$$((x \in \mathbf{N}) \land (y \in \mathbf{N}) \land (c \in \mathbf{N}) \land (\text{iremainder}\,(\text{itimes}\,(c,\ y),\ x) = \mathtt{0}))$$
$$\rightarrow \quad (\text{itimes}\,(x,\ \text{iplus}\,(\text{itimes}\,(c,\ a),\ \text{itimes}\,(b,\ \text{iquotient}\,(\text{itimes}\,(c,\ y),\ x))))$$
$$=\quad \text{itimes}\,(c,\ \text{iplus}\,(\text{itimes}\,(a,\ x),\ \text{itimes}\,(b,\ y))))$$

Theorem: remainder-0-sufficiency

$$((x \in \mathbf{N}) \land (c \in \mathbf{N}) \land (\text{itimes}\,(x,\ p) = c))$$
$$\rightarrow \quad (((c \ \mathbf{mod}\ x) = \mathtt{0}) = \mathbf{t})$$

Theorem: divides-product-reduction

$$((x \in \mathbf{N})$$
$$\land \quad (y \in \mathbf{N})$$
$$\land \quad (c \in \mathbf{N})$$
$$\land \quad (\text{gcd}\,(x,\ y) = \mathtt{1})$$
$$\land \quad (((c * y)\ \mathbf{mod}\ x) = \mathtt{0}))$$
$$\rightarrow \quad ((c\ \mathbf{mod}\ x) = \mathtt{0})$$

Event: Enable remainder-0-sufficiency; name this event ‘remainder-0-sufficiency-off’.

Event: Enable dpr-hack1; name this event ‘dpr-hack1-off’.

Event: Enable dpr-hack2; name this event ‘dpr-hack2-off’.

Event: Enable dpr-hack3; name this event ‘dpr-hack3-off’.

Event: Enable dpr-hack4; name this event ‘dpr-hack4-off’.

Event: Enable dpr-hack5; name this event ‘dpr-hack5-off’.

Event: Enable gcd-factors-gives-linear-combination-rewrite; name this event ‘gcd-factors-gives-linear-combination-rewrite-off’.

Event: Enable gcd-factors-gives-linear-combination; name this event ‘gcd-factors-gives-linear-combination-off’.

Event: Enable gcd-factors; name this event ‘gcd-factors-off’.

Event: Disable idifference-iplus-canonicalizer1; name this event ‘idifference-iplus-canonicalizer1-on’.

Event: Enable ilessp; name this event ‘ilessp-off1’.

Event: Enable idifference; name this event ‘idifference-off1’.

Event: Enable iplus; name this event ‘iplus-off1’.

Event: Enable itimes; name this event ‘itimes-off1’.

Event: Enable ineg; name this event ‘ineg-off1’.

Event: Enable fix-int; name this event ‘fix-int-off1’.

EVENT: Enable izerop; name this event 'izerop-off1'.

EVENT: Enable iremainder; name this event 'iremainder-off1'.

EVENT: Enable iquotient; name this event 'iquotient-off1'.

THEOREM: gcd-remainder-times-fact1-proof
$(\text{gcd}(a,\ b) = 1) \rightarrow ((((b * c) \ \mathbf{mod}\ a) = 0) = ((c \ \mathbf{mod}\ a) = 0))$

THEOREM: times-gcd-fact
$$
\begin{aligned}
&((\text{gcd}(a,\ c) = 1) \wedge (\text{gcd}(b,\ d) = 1)) \\
\rightarrow\quad & (((a * b) = (c * d)) \\
&\quad = \ ((\text{fix}(a) = \text{fix}(d)) \wedge (\text{fix}(b) = \text{fix}(c))))
\end{aligned}
$$

THEOREM: equal-times-times-quotient-arg2
$$
\begin{aligned}
&(((b \ \mathbf{mod}\ e) = 0) \wedge ((d \ \mathbf{mod}\ e) = 0)) \\
\rightarrow\quad & ((((a * (b \div e)) = (c * (d \div e))) \\
&\quad\ = \ ((a * b) = (c * d))) \\
&\quad \wedge \ ((((b \div e) * a) = (c * (d \div e))) \\
&\qquad\ = \ ((a * b) = (c * d))) \\
&\quad \wedge \ ((((b \div e) * a) = ((d \div e) * c)) \\
&\qquad\ = \ ((a * b) = (c * d))))
\end{aligned}
$$

THEOREM: quotient-gcd-times-fact
$$
\begin{aligned}
&(((v * z) = (w * x)) \wedge (0 < w) \wedge (0 < z)) \\
\rightarrow\quad & ((v \div \text{gcd}(v,\ w)) = (x \div \text{gcd}(x,\ z)))
\end{aligned}
$$

THEOREM: quotient-gcd-times-fact1
$$
\begin{aligned}
&(((v * z) = (w * x)) \wedge (0 < w) \wedge (0 < z)) \\
\rightarrow\quad & (((v \div \text{gcd}(v,\ w)) = (x \div \text{gcd}(x,\ z))) = \mathbf{t})
\end{aligned}
$$

THEOREM: quotient-gcd-times-fact2
$$
\begin{aligned}
&(((v * z) = (w * x)) \wedge (0 < w) \wedge (0 < z)) \\
\rightarrow\quad & (((v \div \text{gcd}(w,\ v)) = (x \div \text{gcd}(x,\ z))) = \mathbf{t})
\end{aligned}
$$

THEOREM: quotient-gcd-times-fact3
$$
\begin{aligned}
&(((v * z) = (w * x)) \wedge (0 < w) \wedge (0 < z)) \\
\rightarrow\quad & (((v \div \text{gcd}(v,\ w)) = (x \div \text{gcd}(z,\ x))) = \mathbf{t})
\end{aligned}
$$

THEOREM: quotient-gcd-times-fact4
$$
\begin{aligned}
&(((v * z) = (w * x)) \wedge (0 < w) \wedge (0 < z)) \\
\rightarrow\quad & (((v \div \text{gcd}(w,\ v)) = (x \div \text{gcd}(z,\ x))) = \mathbf{t})
\end{aligned}
$$

THEOREM: quotient-gcd-times-fact5
$(((v * z) = (w * x)) \land (0 < v) \land (0 < x))$
$\rightarrow \quad (((v \div \text{gcd}(v, w)) = (x \div \text{gcd}(x, z))) = \mathbf{t})$

THEOREM: equal-times-gcd-bridge1
$(((b \div \text{gcd}(c, b)) = (d \div \text{gcd}(a, d)))$
$\land \quad ((c \div \text{gcd}(c, b)) = (a \div \text{gcd}(a, d))))$
$\rightarrow \quad (((a * b) = (c * d)) = \mathbf{t})$

THEOREM: numberp-if-integerp-and-not-negativep
$(\text{integerp}(x) \land (\neg\ \text{negativep}(x))) \rightarrow (x \in \mathbf{N})$

THEOREM: negativep-if-integerp-and-not-numberp
$(\text{integerp}(x) \land (x \notin \mathbf{N})) \rightarrow \text{negativep}(x)$

EVENT: Enable numberp-if-integerp-and-not-negativep; name this event 'numberp-if-integerp-and-not-negativep-off'.

EVENT: Enable negativep-if-integerp-and-not-numberp; name this event 'negativep-if-integerp-and-not-numberp-off'.

THEOREM: requal-reduce-reduce-equal
$\text{requal}(a, b) = (\text{reduce}(a) = \text{reduce}(b))$

EVENT: Enable requal-reduce-reduce-equal; name this event 'requal-reduce-reduce-equal-off'.

THEOREM: commutativity-of-requal
$\text{requal}(x, y) = \text{requal}(y, x)$

THEOREM: requal-reduce1
$\text{requal}(\text{reduce}(x), y) = \text{requal}(x, y)$

THEOREM: requal-reduce2
$\text{requal}(x, \text{reduce}(y)) = \text{requal}(x, y)$

THEOREM: reduce-reduce
$\text{reduce}(\text{reduce}(x)) = \text{reduce}(x)$

THEOREM: rplus-open-up
$\text{rplus}(a, b)$
$= \quad$ **if** $\text{rationalp}(a)$
then if $\text{rationalp}(b)$
then $\text{reduce}(\text{rational}(\text{iplus}(\text{itimes}(\text{numerator}(a), \text{denominator}(b)),$

itimes (numerator (b), denominator (a))),
itimes (denominator (a), denominator (b))))
else reduce (fix-rational (a)) **endif**
else reduce (fix-rational (b)) **endif**

Event: Enable rplus-open-up; name this event 'rplus-open-up-off'.

Theorem: commutativity-of-rplus
rplus (x, y) = rplus (y, x)

Theorem: rplus-requal-arg1
requal (a, b) $\rightarrow$ requal (rplus (a, x), rplus (b, x))

Event: Enable rplus-requal-arg1; name this event 'rplus-requal-arg1-off'.

Theorem: requal-x-x
requal (x, x)

Theorem: rplus-reduce-arg1
requal (rplus (reduce (x), y), rplus (x, y))

Theorem: rplus-reduce-arg2
requal (rplus (x, reduce (y)), rplus (x, y))

Theorem: requal-simple-rplus-reduce-arg1
requal (simple-rplus (reduce (x), y), simple-rplus (x, y))

Theorem: requal-simple-rplus-reduce-arg2
requal (simple-rplus (x, reduce (y)), simple-rplus (x, y))

Theorem: reduce-nrationalp
($\neg$ rationalp (x)) $\rightarrow$ (reduce (x) = rational (0, 1))

Theorem: numberp-numerator-reduce
(numerator (reduce (x)) $\in \mathbf{N}$) = (numerator (fix-rational (x)) $\in \mathbf{N}$)

Theorem: negativep-ineg
negativep (ineg (x)) = (0 $<$ x)

Theorem: numberp-ineg
(ineg (x) $\in \mathbf{N}$) = (0 $\not<$ x)

Theorem: rational-ineg-numerator-reduce-bridge
rational (ineg (numerator (reduce (x))), denominator (reduce (x)))
= reduce (rational (ineg (numerator (x)), denominator (x)))

THEOREM: reduce-0
$\text{reduce}\,(\text{rational}\,(\mathtt{0},\ x)) = \text{rational}\,(\mathtt{0},\ \mathtt{1})$

THEOREM: simple-rneg-reduce
$\text{simple-rneg}\,(\text{reduce}\,(x)) = \text{reduce}\,(\text{simple-rneg}\,(x))$

THEOREM: rneg-reduce
$\text{rneg}\,(\text{reduce}\,(x)) = \text{rneg}\,(x)$

THEOREM: simple-rneg-simple-rneg
$\text{requal}\,(\text{simple-rneg}\,(\text{simple-rneg}\,(x)),\ x)$

THEOREM: requal-rneg-rneg
$\text{requal}\,(\text{rneg}\,(\text{rneg}\,(x)),\ x)$

THEOREM: rneg-rneg
$\text{rneg}\,(\text{rneg}\,(x)) = \text{reduce}\,(x)$

THEOREM: rationalp-means
$\text{rationalp}\,(x)$
$\rightarrow\quad (\text{rational-formp}\,(x)$
$\quad\quad \land\quad \text{integerp}\,(\text{numerator}\,(x))$
$\quad\quad \land\quad (\mathtt{0} < \text{denominator}\,(x)))$

THEOREM: means-rationalp
$(\text{integerp}\,(n) \land (\mathtt{0} < d)) \rightarrow \text{rationalp}\,(\text{rational}\,(n,\ d))$

THEOREM: nrational-rplus-arg1
$(\neg\ \text{rationalp}\,(x)) \rightarrow (\text{rplus}\,(x,\ y) = \text{reduce}\,(y))$

THEOREM: nrational-rplus-arg2
$(\neg\ \text{rationalp}\,(x)) \rightarrow (\text{rplus}\,(y,\ x) = \text{reduce}\,(y))$

THEOREM: nrational-simple-rplus-arg1
$(\neg\ \text{rationalp}\,(x)) \rightarrow (\text{simple-rplus}\,(x,\ y) = \text{fix-rational}\,(y))$

THEOREM: nrational-simple-rplus-arg2
$(\neg\ \text{rationalp}\,(x)) \rightarrow (\text{simple-rplus}\,(y,\ x) = \text{fix-rational}\,(y))$

THEOREM: simple-rplus-fix-rational-arg1
$\text{simple-rplus}\,(\text{fix-rational}\,(x),\ y) = \text{simple-rplus}\,(x,\ y)$

THEOREM: simple-rplus-fix-rational-arg2
$\text{simple-rplus}\,(x,\ \text{fix-rational}\,(y)) = \text{simple-rplus}\,(x,\ y)$

THEOREM: fix-rational-of-rationalp
$\text{rationalp}\,(x) \rightarrow (\text{fix-rational}\,(x) = x)$

THEOREM: negative-guts-ineg
negative-guts (ineg (x))
$=$ **if** $0 < x$ **then** x
else 0 **endif**

THEOREM: rationalp-simple-rplus
rationalp (simple-rplus (x, y))

THEOREM: fix-rational-simple-rplus
fix-rational (simple-rplus (x, y)) $=$ simple-rplus (x, y)

THEOREM: requal-associativity-of-simple-rplus
requal (simple-rplus (simple-rplus (x, y), z), simple-rplus (x, simple-rplus (y, z)))

THEOREM: equal-times-bridge1
$(((a * b) = (c * d)) \land ((a * x) = (c * y)) \land (a \not\simeq 0))$
$\rightarrow \quad (((b * y) = (d * x)) = \mathbf{t})$

THEOREM: equal-times-bridge2
$(((a * b) = (c * d)) \land ((a * x) = (c * y)) \land (a \not\simeq 0))$
$\rightarrow \quad (((y * b) = (d * x)) = \mathbf{t})$

THEOREM: equal-times-bridge3
$(((a * b) = (c * d)) \land ((a * x) = (c * y)) \land (a \not\simeq 0))$
$\rightarrow \quad (((b * y) = (x * d)) = \mathbf{t})$

THEOREM: equal-times-bridge4
$(((a * b) = (c * d)) \land ((a * x) = (c * y)) \land (a \not\simeq 0))$
$\rightarrow \quad (((y * b) = (x * d)) = \mathbf{t})$

THEOREM: transitivity-of-requal-bridge
(requal (a, b) $\land$ requal (b, c)) $\rightarrow$ requal (a, c)

EVENT: Enable transitivity-of-requal-bridge; name this event 'transitivity-of-requal-bridge-off'.

THEOREM: transitivity-of-requal
((requal (a, b) $\land$ requal (b, c)) $\rightarrow$ requal (a, c))
$\land$ ((requal (a, b) $\land$ requal (c, b)) $\rightarrow$ requal (a, c))
$\land$ ((requal (b, a) $\land$ requal (b, c)) $\rightarrow$ requal (a, c))
$\land$ ((requal (b, a) $\land$ requal (c, b)) $\rightarrow$ requal (a, c))

THEOREM: equal-requal-rewrite
(requal (b, c) $\rightarrow$ (requal (a, b) $=$ requal (a, c)))
$\land$ (requal (b, c) $\rightarrow$ (requal (a, b) $=$ requal (c, a)))
$\land$ (requal (b, c) $\rightarrow$ (requal (b, a) $=$ requal (c, a)))

EVENT: Enable equal-requal-rewrite; name this event 'equal-requal-rewrite-off'.

THEOREM: requal-simple-rplus-bridge

$(\text{requal}(\text{simple-rplus}(\text{reduce}(x),\ y),\ z) = \text{requal}(\text{simple-rplus}(x,\ y),\ z))$
$\land\ \ (\text{requal}(\text{simple-rplus}(x,\ \text{reduce}(y)),\ z) = \text{requal}(\text{simple-rplus}(x,\ y),\ z))$
$\land\ \ (\text{requal}(z,\ \text{simple-rplus}(\text{reduce}(x),\ y)) = \text{requal}(z,\ \text{simple-rplus}(x,\ y)))$
$\land\ \ (\text{requal}(z,\ \text{simple-rplus}(x,\ \text{reduce}(y))) = \text{requal}(z,\ \text{simple-rplus}(x,\ y)))$

THEOREM: requal-associativity-of-rplus

$\text{requal}(\text{rplus}(\text{rplus}(x,\ y),\ z),\ \text{rplus}(x,\ \text{rplus}(y,\ z)))$

THEOREM: associativity-of-rplus

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

THEOREM: commutativity2-of-rplus

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

THEOREM: rplus-rdifference-arg1

$\text{rplus}(\text{rdifference}(x,\ y),\ z) = \text{rdifference}(\text{rplus}(x,\ z),\ y)$

THEOREM: rplus-rdifference-arg2

$\text{rplus}(x,\ \text{rdifference}(y,\ z)) = \text{rdifference}(\text{rplus}(x,\ y),\ z)$

THEOREM: reduce-rplus

$\text{reduce}(\text{rplus}(x,\ y)) = \text{rplus}(x,\ y)$

THEOREM: reduce-rtimes

$\text{reduce}(\text{rtimes}(x,\ y)) = \text{rtimes}(x,\ y)$

THEOREM: reduce-difference

$\text{reduce}(\text{rdifference}(x,\ y)) = \text{rdifference}(x,\ y)$

THEOREM: reduce-rquotient

$\text{reduce}(\text{rquotient}(x,\ y)) = \text{rquotient}(x,\ y)$

THEOREM: reduce-rmagnitude

$\text{reduce}(\text{rmagnitude}(x)) = \text{rmagnitude}(x)$

THEOREM: reduce-rneg

$\text{reduce}(\text{rneg}(x)) = \text{rneg}(x)$

THEOREM: rplus-reduce-arg1-rewrite

$\text{rplus}(\text{reduce}(x),\ y) = \text{rplus}(x,\ y)$

THEOREM: rplus-reduce-arg2-rewrite

$\text{rplus}(x,\ \text{reduce}(y)) = \text{rplus}(x,\ y)$

THEOREM: requal-rneg-simple-rplus
requal (simple-rplus (simple-rneg (x), simple-rneg (y)),
simple-rneg (simple-rplus (x, y)))

THEOREM: requal-rplus-rneg
requal (rneg (rplus (x, y)), rplus (rneg (x), rneg (y)))

THEOREM: rneg-rplus
rneg (rplus (x, y)) = rplus (rneg (x), rneg (y))

THEOREM: rdifference-rdifference-arg1
rdifference (rdifference (x, y), z) = rdifference (x, rplus (y, z))

THEOREM: rdifference-rdifference-arg2
rdifference (x, rdifference (y, z)) = rdifference (rplus (x, z), y)

THEOREM: rplus-rneg-arg1
rplus (rneg (x), y) = rdifference (y, x)

THEOREM: rplus-rneg-arg2
rplus (x, rneg (y)) = rdifference (x, y)

THEOREM: commutativity-of-simple-rtimes
simple-rtimes (x, y) = simple-rtimes (y, x)

THEOREM: commutativity-of-rtimes
rtimes (x, y) = rtimes (y, x)

THEOREM: simple-rtimes-rzerop
rzerop (x)
$\rightarrow$ ((numerator (simple-rtimes (x, y)) = 0)
$\land$ (numerator (simple-rtimes (y, x)) = 0))

THEOREM: associativity-of-simple-rtimes-when-not-rzerop
(($\neg$ rzerop (x)) $\land$ ($\neg$ rzerop (y)) $\land$ ($\neg$ rzerop (z)))
$\rightarrow$ requal (simple-rtimes (simple-rtimes (x, y), z),
simple-rtimes (x, simple-rtimes (y, z)))

THEOREM: numerator-zero-rzerop-bridge
(numerator (x) = 0) $\rightarrow$ rzerop (x)

THEOREM: associativity-of-simple-rtimes
requal (simple-rtimes (simple-rtimes (x, y), z),
simple-rtimes (x, simple-rtimes (y, z)))

THEOREM: requal-simple-rtimes-requal-arg1
requal (x, y) $\rightarrow$ requal (simple-rtimes (x, z), simple-rtimes (y, z))

THEOREM: requal-simple-rtimes-requal-arg2
$\text{requal}\,(x,\ y) \rightarrow \text{requal}\,(\text{simple-rtimes}\,(z,\ x),\ \text{simple-rtimes}\,(z,\ y))$

THEOREM: requal-simple-rtimes-reduce-arg1
$\text{requal}\,(\text{simple-rtimes}\,(\text{reduce}\,(x),\ y),\ \text{simple-rtimes}\,(x,\ y))$

THEOREM: requal-simple-rtimes-reduce-arg2
$\text{requal}\,(\text{simple-rtimes}\,(x,\ \text{reduce}\,(y)),\ \text{simple-rtimes}\,(x,\ y))$

THEOREM: requal-simple-rtimes-bridge

$$
\begin{aligned}
&(\text{requal}\,(\text{simple-rtimes}\,(\text{reduce}\,(x),\ y),\ z) = \text{requal}\,(\text{simple-rtimes}\,(x,\ y),\ z))\\
\land\quad &(\text{requal}\,(\text{simple-rtimes}\,(x,\ \text{reduce}\,(y)),\ z)\\
&\quad = \quad \text{requal}\,(\text{simple-rtimes}\,(x,\ y),\ z))\\
\land\quad &(\text{requal}\,(z,\ \text{simple-rtimes}\,(\text{reduce}\,(x),\ y))\\
&\quad = \quad \text{requal}\,(z,\ \text{simple-rtimes}\,(x,\ y)))\\
\land\quad &(\text{requal}\,(z,\ \text{simple-rtimes}\,(x,\ \text{reduce}\,(y)))\\
&\quad = \quad \text{requal}\,(z,\ \text{simple-rtimes}\,(x,\ y)))
\end{aligned}
$$

THEOREM: requal-associativity-of-rtimes
$\text{requal}\,(\text{rtimes}\,(\text{rtimes}\,(x,\ y),\ z),\ \text{rtimes}\,(x,\ \text{rtimes}\,(y,\ z)))$

THEOREM: associativity-of-rtimes
$\text{rtimes}\,(\text{rtimes}\,(x,\ y),\ z) = \text{rtimes}\,(x,\ \text{rtimes}\,(y,\ z))$

THEOREM: simple-rplus-rzerop
$\text{rzerop}\,(x) \rightarrow (\text{requal}\,(\text{simple-rplus}\,(x,\ y),\ y) \land \text{requal}\,(\text{simple-rplus}\,(y,\ x),\ y))$

THEOREM: numerator-type
$(\text{numerator}\,(x) \in \mathbf{N}) \lor \text{negativep}\,(\text{numerator}\,(x))$

THEOREM: rationalp-simple-rtimes
$\text{rationalp}\,(\text{simple-rtimes}\,(x,\ y))$

THEOREM: fix-rational-simple-rtimes
$\text{fix-rational}\,(\text{simple-rtimes}\,(x,\ y)) = \text{simple-rtimes}\,(x,\ y)$

THEOREM: requal-simple-rtimes-simple-rplus-when-not-rzerop

$$
\begin{aligned}
&((\neg\ \text{rzerop}\,(x)) \land (\neg\ \text{rzerop}\,(y)) \land (\neg\ \text{rzerop}\,(z)))\\
\rightarrow\quad &\text{requal}\,(\text{simple-rtimes}\,(\text{simple-rplus}\,(x,\ y),\ z),\\
&\qquad \text{simple-rplus}\,(\text{simple-rtimes}\,(x,\ z),\ \text{simple-rtimes}\,(y,\ z)))
\end{aligned}
$$

THEOREM: requal-rzerop-simple-rplus

$$
\begin{aligned}
&\text{rzerop}\,(x)\\
\rightarrow\quad &(\text{requal}\,(\text{simple-rplus}\,(x,\ y),\ \text{fix-rational}\,(y))\\
&\ \land\quad \text{requal}\,(\text{simple-rplus}\,(y,\ x),\ \text{fix-rational}\,(y)))
\end{aligned}
$$

THEOREM: commutativity-of-simple-rplus
simple-rplus $(x,\ y)$ = simple-rplus $(y,\ x)$

THEOREM: simple-rplus-bridge
(rzerop (x) $\vee$ rzerop (y) $\vee$ rzerop (z))
$\rightarrow$ requal (simple-rtimes (simple-rplus $(x,\ y)$, z),
simple-rplus (simple-rtimes $(x,\ z)$, simple-rtimes $(y,\ z)$)))

THEOREM: requal-simple-rtimes-simple-rplus-arg1
requal (simple-rtimes (simple-rplus $(x,\ y)$, z),
simple-rplus (simple-rtimes $(x,\ z)$, simple-rtimes $(y,\ z)$)))

THEOREM: requal-rtimes-rplus-bridge
requal (rtimes (rplus $(x,\ y)$, z), rplus (rtimes $(x,\ z)$, rtimes $(y,\ z)$)))

THEOREM: rtimes-rplus-arg1
rtimes (rplus $(x,\ y)$, z) = rplus (rtimes $(x,\ z)$, rtimes $(y,\ z)$)

THEOREM: rtimes-rplus-arg2
rtimes (x, rplus $(y,\ z)$) = rplus (rtimes $(x,\ y)$, rtimes $(x,\ z)$)

THEOREM: requal-simple-rtimes-simple-rneg
requal (simple-rtimes (simple-rneg (x), y), simple-rneg (simple-rtimes $(x,\ y)$)))

THEOREM: requal-rtimes-rneg
requal (rtimes (rneg (x), y), rneg (rtimes $(x,\ y)$)))

THEOREM: rtimes-rneg-arg1
rtimes (rneg (x), y) = rneg (rtimes $(x,\ y)$)

THEOREM: rtimes-rneg-arg2
rtimes (x, rneg (y)) = rneg (rtimes $(x,\ y)$)

THEOREM: rtimes-rdifference-arg1
rtimes (rdifference $(x,\ y)$, z) = rdifference (rtimes $(x,\ z)$, rtimes $(y,\ z)$)

THEOREM: rtimes-rdifference-arg2
rtimes (x, rdifference $(y,\ z)$) = rdifference (rtimes $(x,\ y)$, rtimes $(x,\ z)$)

THEOREM: rneg-rdifference
rneg (rdifference $(x,\ y)$) = rdifference $(y,\ x)$

THEOREM: rdifference-rneg-arg2
rdifference (x, rneg (y)) = rplus $(x,\ y)$

EVENT: Enable rdifference-rneg-arg2; name this event ‘rdifference-rneg-arg2-off’.

EVENT: Enable rneg-rdifference; name this event ‘rneg-rdifference-off’.

EVENT: Enable rtimes-rdifference-arg2; name this event ‘rtimes-rdifference-arg2-off’.

EVENT: Enable rtimes-rdifference-arg1; name this event ‘rtimes-rdifference-arg1-off’.

EVENT: Enable rtimes-rneg-arg2; name this event ‘rtimes-rneg-arg2-off’.

EVENT: Enable rtimes-rneg-arg1; name this event ‘rtimes-rneg-arg1-off’.

EVENT: Enable requal-rtimes-rneg; name this event ‘requal-rtimes-rneg-off’.

EVENT: Enable requal-simple-rtimes-simple-rneg; name this event ‘requal-simple-rtimes-simple-rneg-off’.

EVENT: Enable rtimes-rplus-arg2; name this event ‘rtimes-rplus-arg2-off’.

EVENT: Enable rtimes-rplus-arg1; name this event ‘rtimes-rplus-arg1-off’.

EVENT: Enable requal-rtimes-rplus-bridge; name this event ‘requal-rtimes-rplus-bridge-off’.

EVENT: Enable requal-simple-rtimes-simple-rplus-arg1; name this event ‘requal-simple-rtimes-simple-rplus-arg1-off’.

EVENT: Enable simple-rplus-bridge; name this event ‘simple-rplus-bridge-off’.

EVENT: Enable commutativity-of-simple-rplus; name this event ‘commutativity-of-simple-rplus-off’.

EVENT: Enable requal-rzerop-simple-rplus; name this event ‘requal-rzerop-simple-rplus-off’.

EVENT: Enable requal-simple-rtimes-simple-rplus-when-not-rzerop; name this event ‘requal-simple-rtimes-simple-rplus-when-not-rzerop-off’.

EVENT: Enable fix-rational-simple-rtimes; name this event ‘fix-rational-simple-rtimes-off’.

EVENT: Enable rationalp-simple-rtimes; name this event ‘rationalp-simple-rtimes-off’.

EVENT: Enable numerator-type; name this event ‘numerator-type-off’.

EVENT: Enable simple-rplus-rzerop; name this event ‘simple-rplus-rzerop-off’.

EVENT: Enable associativity-of-rtimes; name this event ‘associativity-of-rtimes-off’.

EVENT: Enable requal-associativity-of-rtimes; name this event ‘requal-associativity-of-rtimes-off’.

EVENT: Enable requal-simple-rtimes-bridge; name this event ‘requal-simple-rtimes-bridge-off’.

EVENT: Enable requal-simple-rtimes-reduce-arg2; name this event ‘requal-simple-rtimes-reduce-arg2-off’.

EVENT: Enable requal-simple-rtimes-reduce-arg1; name this event ‘requal-simple-rtimes-reduce-arg1-off’.

EVENT: Enable requal-simple-rtimes-requal-arg2; name this event ‘requal-simple-rtimes-requal-arg2-off’.

EVENT: Enable requal-simple-rtimes-requal-arg1; name this event ‘requal-simple-rtimes-requal-arg1-off’.

EVENT: Enable associativity-of-simple-rtimes; name this event ‘associativity-of-simple-rtimes-off’.

EVENT: Enable numerator-zero-rzerop-bridge; name this event 'numerator-zero-rzerop-bridge-off'.

EVENT: Enable associativity-of-simple-rtimes-when-not-rzerop; name this event 'associativity-of-simple-rtimes-when-not-rzerop-off'.

EVENT: Enable simple-rtimes-rzerop; name this event 'simple-rtimes-rzerop-off'.

EVENT: Enable rzerop; name this event 'rzerop-off'.

EVENT: Enable commutativity-of-rtimes; name this event 'commutativity-of-rtimes-off'.

EVENT: Enable commutativity-of-simple-rtimes; name this event 'commutativity-of-simple-rtimes-off'.

EVENT: Enable rplus-rneg-arg2; name this event 'rplus-rneg-arg2-off'.

EVENT: Enable rplus-rneg-arg1; name this event 'rplus-rneg-arg1-off'.

EVENT: Enable rdifference-rdifference-arg2; name this event 'rdifference-rdifference-arg2-off'.

EVENT: Enable rdifference-rdifference-arg1; name this event 'rdifference-rdifference-arg1-off'.

EVENT: Enable rneg-rplus; name this event 'rneg-rplus-off'.

EVENT: Enable requal-rplus-rneg; name this event 'requal-rplus-rneg-off'.

EVENT: Enable requal-rneg-simple-rplus; name this event 'requal-rneg-simple-rplus-off'.

EVENT: Enable rplus-reduce-arg2-rewrite; name this event 'rplus-reduce-arg2-rewrite-off'.

EVENT: Enable rplus-reduce-arg1-rewrite; name this event 'rplus-reduce-arg1-rewrite-off'.

EVENT: Enable reduce-rneg; name this event 'reduce-rneg-off'.

EVENT: Enable reduce-rmagnitude; name this event 'reduce-rmagnitude-off'.

EVENT: Enable reduce-rquotient; name this event 'reduce-rquotient-off'.

EVENT: Enable reduce-difference; name this event 'reduce-difference-off'.

EVENT: Enable reduce-rtimes; name this event 'reduce-rtimes-off'.

EVENT: Enable reduce-rplus; name this event 'reduce-rplus-off'.

EVENT: Enable rplus-rdifference-arg2; name this event 'rplus-rdifference-arg2-off'.

EVENT: Enable rplus-rdifference-arg1; name this event 'rplus-rdifference-arg1-off'.

EVENT: Enable commutativity2-of-rplus; name this event 'commutativity2-of-rplus-off'.

EVENT: Enable associativity-of-rplus; name this event 'associativity-of-rplus-off'.

EVENT: Enable requal-associativity-of-rplus; name this event 'requal-associativity-of-rplus-off'.

EVENT: Enable requal-simple-rplus-bridge; name this event 'requal-simple-rplus-bridge-off'.

EVENT: Enable equal-requal-rewrite; name this event 'equal-requal-rewrite-off1'.

EVENT: Enable transitivity-of-requal; name this event 'transitivity-of-requal-

off'.

Event: Enable transitivity-of-requal-bridge; name this event 'transitivity-of-requal-bridge-off1'.

Event: Enable equal-times-bridge4; name this event 'equal-times-bridge4-off'.

Event: Enable equal-times-bridge3; name this event 'equal-times-bridge3-off'.

Event: Enable equal-times-bridge2; name this event 'equal-times-bridge2-off'.

Event: Enable equal-times-bridge1; name this event 'equal-times-bridge1-off'.

Event: Enable requal-associativity-of-simple-rplus; name this event 'requal-associativity-of-simple-rplus-off'.

Event: Enable fix-rational-simple-rplus; name this event 'fix-rational-simple-rplus-off'.

Event: Enable rationalp-simple-rplus; name this event 'rationalp-simple-rplus-off'.

Event: Enable negative-guts-ineg; name this event 'negative-guts-ineg-off'.

Event: Enable fix-rational-of-rationalp; name this event 'fix-rational-of-rationalp-off'.

Event: Enable simple-rplus-fix-rational-arg2; name this event 'simple-rplus-fix-rational-arg2-off'.

Event: Enable simple-rplus-fix-rational-arg1; name this event 'simple-rplus-fix-rational-arg1-off'.

Event: Enable nrational-simple-rplus-arg2; name this event 'nrational-simple-rplus-arg2-off'.

Event: Enable nrational-simple-rplus-arg1; name this event 'nrational-simple-rplus-arg1-off'.

EVENT: Enable nrational-rplus-arg2; name this event 'nrational-rplus-arg2-off'.

EVENT: Enable nrational-rplus-arg1; name this event 'nrational-rplus-arg1-off'.

EVENT: Enable means-rationalp; name this event 'means-rationalp-off'.

EVENT: Enable rationalp-means; name this event 'rationalp-means-off'.

EVENT: Enable rneg-rneg; name this event 'rneg-rneg-off'.

EVENT: Enable requal-rneg-rneg; name this event 'requal-rneg-rneg-off'.

EVENT: Enable simple-rneg-simple-rneg; name this event 'simple-rneg-simple-rneg-off'.

EVENT: Enable rneg-reduce; name this event 'rneg-reduce-off'.

EVENT: Enable simple-rneg-reduce; name this event 'simple-rneg-reduce-off'.

EVENT: Enable reduce-0; name this event 'reduce-0-off'.

EVENT: Enable rational-ineg-numerator-reduce-bridge; name this event 'rational-ineg-numerator-reduce-bridge-off'.

EVENT: Enable numberp-ineg; name this event 'numberp-ineg-off'.

EVENT: Enable negativep-ineg; name this event 'negativep-ineg-off'.

EVENT: Enable numberp-numerator-reduce; name this event 'numberp-numerator-reduce-off'.

EVENT: Enable reduce-nrationalp; name this event 'reduce-nrationalp-off'.

EVENT: Enable requal-simple-rplus-reduce-arg2; name this event 'requal-simple-rplus-reduce-arg2-off'.

EVENT: Enable requal-simple-rplus-reduce-arg1; name this event 'requal-simple-rplus-reduce-arg1-off'.

EVENT: Enable rplus-reduce-arg2; name this event 'rplus-reduce-arg2-off'.

EVENT: Enable rplus-reduce-arg1; name this event 'rplus-reduce-arg1-off'.

EVENT: Enable requal-x-x; name this event 'requal-x-x-off'.

EVENT: Enable rplus-requal-arg1; name this event 'rplus-requal-arg1-off1'.

EVENT: Enable commutativity-of-rplus; name this event 'commutativity-of-rplus-off'.

EVENT: Enable rplus-open-up; name this event 'rplus-open-up-off1'.

EVENT: Enable reduce-reduce; name this event 'reduce-reduce-off'.

EVENT: Enable requal-reduce2; name this event 'requal-reduce2-off'.

EVENT: Enable requal-reduce1; name this event 'requal-reduce1-off'.

EVENT: Enable commutativity-of-requal; name this event 'commutativity-of-requal-off'.

EVENT: Enable requal-reduce-reduce-equal; name this event 'requal-reduce-reduce-equal-off1'.

EVENT: Enable negativep-if-integerp-and-not-numberp; name this event 'negativep-if-integerp-and-not-numberp-off1'.

EVENT: Enable numberp-if-integerp-and-not-negativep; name this event 'numberp-if-integerp-and-not-negativep-off1'.

EVENT: Enable equal-times-gcd-bridge1; name this event ‘equal-times-gcd-bridge1-off’.

EVENT: Enable quotient-gcd-times-fact5; name this event ‘quotient-gcd-times-fact5-off’.

EVENT: Enable quotient-gcd-times-fact4; name this event ‘quotient-gcd-times-fact4-off’.

EVENT: Enable quotient-gcd-times-fact3; name this event ‘quotient-gcd-times-fact3-off’.

EVENT: Enable quotient-gcd-times-fact2; name this event ‘quotient-gcd-times-fact2-off’.

EVENT: Enable quotient-gcd-times-fact1; name this event ‘quotient-gcd-times-fact1-off’.

EVENT: Enable quotient-gcd-times-fact; name this event ‘quotient-gcd-times-fact-off’.

EVENT: Enable equal-times-times-quotient-arg2; name this event ‘equal-times-times-quotient-arg2-off’.

EVENT: Enable times-gcd-fact; name this event ‘times-gcd-fact-off’.

EVENT: Enable gcd-remainder-times-fact1-proof; name this event ‘gcd-remainder-times-fact1-proof-off’.

EVENT: Enable divides-product-reduction; name this event ‘divides-product-reduction-off’.

EVENT: Enable remainder-0-sufficiency; name this event ‘remainder-0-sufficiency-off1’.

EVENT: Enable dpr-hack5; name this event ‘dpr-hack5-off1’.

EVENT: Enable dpr-hack4; name this event ‘dpr-hack4-off1’.

EVENT: Enable dpr-hack3; name this event ‘dpr-hack3-off1’.

EVENT: Enable dpr-hack2; name this event ‘dpr-hack2-off1’.

EVENT: Enable dpr-hack1; name this event ‘dpr-hack1-off1’.

EVENT: Enable gcd-factors-gives-linear-combination-rewrite; name this event ‘gcd-factors-gives-linear-combination-rewrite-off1’.

EVENT: Enable gcd-factors-gives-linear-combination; name this event ‘gcd-factors-gives-linear-combination-off1’.

EVENT: Enable gcd-factors; name this event ‘gcd-factors-off1’.

EVENT: Enable divides-each-equality; name this event ‘divides-each-equality-off’.

EVENT: Enable gcd-quotient-quotient; name this event ‘gcd-quotient-quotient-off’.

EVENT: Enable gcd-times2; name this event ‘gcd-times2-off’.

EVENT: Enable gcd-times1; name this event ‘gcd-times1-off’.

EVENT: Enable gcd-times1-induct; name this event ‘gcd-times1-induct-off’.

EVENT: Enable gcd-remainder-fact2; name this event ‘gcd-remainder-fact2-off’.

EVENT: Enable gcd-remainder-fact1; name this event ‘gcd-remainder-fact1-off’.

EVENT: Enable rational-generalization; name this event ‘rational-generalization-off’.

EVENT: Enable fix-rational-rmagnitude; name this event ‘fix-rational-rmagnitude-

off'.

EVENT: Enable fix-rational-rquotient; name this event 'fix-rational-rquotient-off'.

EVENT: Enable fix-rational-rdifference; name this event 'fix-rational-rdifference-off'.

EVENT: Enable fix-rational-rneg; name this event 'fix-rational-rneg-off'.

EVENT: Enable fix-rational-rtimes; name this event 'fix-rational-rtimes-off'.

EVENT: Enable fix-rational-fix-rational; name this event 'fix-rational-fix-rational-off'.

EVENT: Enable fix-rational-rplus; name this event 'fix-rational-rplus-off'.

EVENT: Enable fix-rational-reduce; name this event 'fix-rational-reduce-off'.

EVENT: Enable rationalp-rmagnitude; name this event 'rationalp-rmagnitude-off'.

EVENT: Enable rationalp-rquotient; name this event 'rationalp-rquotient-off'.

EVENT: Enable rationalp-rdifference; name this event 'rationalp-rdifference-off'.

EVENT: Enable rationalp-rneg; name this event 'rationalp-rneg-off'.

EVENT: Enable rationalp-rtimes; name this event 'rationalp-rtimes-off'.

EVENT: Enable rationalp-fix-rational; name this event 'rationalp-fix-rational-off'.

EVENT: Enable rationalp-rplus; name this event 'rationalp-rplus-off'.

EVENT: Enable rationalp-reduce; name this event 'rationalp-reduce-off'.

EVENT: Enable iplus-is-plus; name this event 'iplus-is-plus-off'.

EVENT: Enable itimes-is-times; name this event 'itimes-is-times-off'.

EVENT: Enable equal-ineg-ineg; name this event 'equal-ineg-ineg-off'.

EVENT: Enable itimes-negativep-arg2; name this event 'itimes-negativep-arg2-off'.

EVENT: Enable itimes-negativep-arg1; name this event 'itimes-negativep-arg1-off'.

EVENT: Enable integerp-if-numberp; name this event 'integerp-if-numberp-off'.

EVENT: Enable integerp-if-negativep-non-zero; name this event 'integerp-if-negativep-non-zero-off'.

EVENT: Enable itimes-ineg-arg2; name this event 'itimes-ineg-arg2-off'.

EVENT: Enable itimes-ineg-arg1; name this event 'itimes-ineg-arg1-off'.

EVENT: Enable fix-int-on-integers; name this event 'fix-int-on-integers-off'.

EVENT: Enable integerp-minus; name this event 'integerp-minus-off'.

EVENT: Enable requal; name this event 'requal-off'.

EVENT: Enable rlessp; name this event 'rlessp-off'.

EVENT: Enable rmagnitude; name this event 'rmagnitude-off'.

EVENT: Enable simple-rmagnitude; name this event 'simple-rmagnitude-off'.

EVENT: Enable rquotient; name this event 'rquotient-off'.

EVENT: Enable rtimes; name this event 'rtimes-off'.

EVENT: Enable simple-rtimes; name this event ‘simple-rtimes-off’.

EVENT: Enable rdifference; name this event ‘rdifference-off’.

EVENT: Enable rneg; name this event ‘rneg-off’.

EVENT: Enable simple-rneg; name this event ‘simple-rneg-off’.

EVENT: Enable rplus; name this event ‘rplus-off’.

EVENT: Enable simple-rplus; name this event ‘simple-rplus-off’.

EVENT: Enable reduce; name this event ‘reduce-off’.

EVENT: Enable fix-rational; name this event ‘fix-rational-off’.

EVENT: Enable rationalp; name this event ‘rationalp-off’.

EVENT: Let us define the theory *r1* to consist of the following events: rdifference-rneg-arg2, rneg-rdifference, rtimes-rdifference-arg2, rtimes-rdifference-arg1, rtimes-rneg-arg2, rtimes-rneg-arg1, rtimes-rplus-arg2, rtimes-rplus-arg1, associativity-of-rtimes, rzerop, commutativity-of-rtimes, rplus-rneg-arg2, rplus-rneg-arg1, rdifference-rdifference-arg2, rdifference-rdifference-arg1, rneg-rplus, rplus-reduce-arg2-rewrite, rplus-reduce-arg1-rewrite, reduce-rneg, reduce-rmagnitude, reduce-rquotient, reduce-difference, reduce-rtimes, reduce-rplus, rplus-rdifference-arg2, rplus-rdifference-arg1, commutativity2-of-rplus, associativity-of-rplus, equal-requal-rewrite, transitivity-of-requal, fix-rational-of-rationalp, nrational-rplus-arg2, nrational-rplus-arg1, rationalp-means, means-rationalp, rneg-rneg, rneg-reduce, reduce-0, numberp-numerator-reduce, reduce-nrationalp, rplus-reduce-arg2, rplus-reduce-arg1, requal-x-x, rplus-requal-arg1, commutativity-of-rplus, reduce-reduce, requal-reduce2, requal-reduce1, commutativity-of-requal, rational-generalization, fix-rational-rmagnitude, fix-rational-rquotient, fix-rational-rdifference, fix-rational-rneg, fix-rational-rtimes, fix-rational-fix-rational, fix-rational-rplus, fix-rational-reduce, rationalp-rmagnitude, rationalp-rquotient, rationalp-rdifference, rationalp-rneg, rationalp-rtimes, rationalp-fix-rational, rationalp-rplus, rationalp-reduce.

DEFINITION:
rplus-tree (x)
= **if** $x \simeq$ **nil then** `'(rational '0 '1)`

elseif cdr $(x) \simeq$ **nil** **then** list (`'reduce`, car (x))
elseif cddr $(x) \simeq$ **nil** **then** list (`'rplus`, car (x), cadr (x))
else list (`'rplus`, car (x), rplus-tree (cdr (x))) **endif**

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

DEFINITION:
split-by-parity (x)
= **if** listp (x)
then if (caar (x) = `'rneg`)
$\wedge$ (caadar $(x) \neq$ `'rneg`)
$\wedge$ (cddar (x) = **nil**)
then cons (car (split-by-parity (cdr (x))),
cons (cadar (x), cdr (split-by-parity (cdr (x)))))
else cons (cons (car (x), car (split-by-parity (cdr (x)))),
cdr (split-by-parity (cdr (x)))) **endif**
else `'(nil)` **endif**

DEFINITION:
make-negs (x)
= **if** listp (x) **then** cons (list (`'rneg`, car (x)), make-negs (cdr (x)))
else nil endif

THEOREM: rplus-rzerop-bridge
$((v \not\simeq 0) \wedge (z \in \mathbf{N}))$
$\rightarrow$ (reduce (rational $(v * z,\ v * w)$) = reduce (rational $(z,\ w)$))

THEOREM: rplus-rzerop-bridge2
$(v \not\simeq 0)$
$\rightarrow$ (reduce (rational $(-\ (d * v),\ v * w)$)
= reduce (rational $(-\ d,\ w)$))

THEOREM: rplus-rzerop
rzerop $(x) \rightarrow$ ((rplus $(x,\ y)$ = reduce (y)) $\wedge$ (rplus $(y,\ x)$ = reduce (y)))

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

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

THEOREM: reduce-eval$-rplus-tree
reduce (eval$ ($\mathbf{t}$, rplus-tree (y), a)) = eval$ ($\mathbf{t}$, rplus-tree (y), a)

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

THEOREM: member-append
($a \in$ append (x, y)) = (($a \in x$) $\vee$ ($a \in y$))

THEOREM: delete-append
delete (x, append (a, b))
= **if** $x \in a$ **then** append (delete (x, a), b)
else append (a, delete (x, b)) **endif**

THEOREM: member-subbagp
(($a \in x$) $\wedge$ subbagp (x, y)) $\rightarrow$ ($a \in y$)

DEFINITION:
badguy (x, y)
= **if** listp (x)
then if car (x) $\in y$ **then** badguy (cdr (x), delete (car (x), y))
else car (x) **endif**
else 0 **endif**

THEOREM: member-occur
($a \in x$) = (0 < occurrences (a, x))

THEOREM: occurrences-delete2
occurrences (a, delete (b, x))
= **if** $a = b$ **then** occurrences (a, x) $-$ 1
else occurrences (a, x) **endif**

THEOREM: subbagp-wit-lemma
subbagp (x, y)
= (occurrences (badguy (x, y), y) $\not<$ occurrences (badguy (x, y), x))

THEOREM: occurrences-append
occurrences (a, append (x, y)) = (occurrences (a, x) + occurrences (a, y))

THEOREM: subbagp-append
subbagp (append (x, y), append (y, x))

EVENT: Enable member-occur; name this event 'member-occur-off'.

EVENT: Enable subbagp-wit-lemma; name this event 'subbagp-wit-lemma-off'.

THEOREM: subbagp-delete-same
subbagp $(x,\ y) \rightarrow$ subbagp (delete $(a,\ x)$, delete $(a,\ y)$)

THEOREM: subbagp-delete-same-means
$((a \in y) \wedge$ subbagp (delete $(a,\ x)$, delete $(a,\ y))) \rightarrow$ subbagp $(x,\ y)$

THEOREM: subbagp-delete-car
subbagp $(x,\ y) \rightarrow$ subbagp (delete (car (y), x), cdr (y))

THEOREM: subbagp-delete-car2
subbagp $(x,\ y) \rightarrow$ subbagp (cdr (x), delete (car (x), y))

THEOREM: subbagp-permutation
(subbagp $(x,\ y) \wedge$ subbagp $(y,\ x)) \rightarrow$ permutation $(x,\ y)$

THEOREM: permutation-a-b
permutation (append $(a,\ b)$, append $(b,\ a)$)

THEOREM: not-subbagp-not-permutation
$(\neg$ subbagp $(x,\ y)) \rightarrow ((\neg$ permutation $(x,\ y)) \wedge (\neg$ permutation $(y,\ x)))$

THEOREM: permutation-as-subbagp-helper
permutation $(x,\ y) \leftrightarrow$ (subbagp $(x,\ y) \wedge$ subbagp $(y,\ x)$)

THEOREM: permutation-as-subbagp
permutation $(x,\ y) =$ (subbagp $(x,\ y) \wedge$ subbagp $(y,\ x)$)

EVENT: Enable permutation-as-subbagp-helper; name this event 'permutation-as-subbagp-helper-off'.

EVENT: Enable permutation-as-subbagp; name this event 'permutation-as-subbagp-off'.

THEOREM: subbagp-necc
subbagp $(x,\ y) \rightarrow$ (occurrences $(a,\ y) \not< $ occurrences $(a,\ x)$)

THEOREM: subbagp-transitive
(subbagp $(x,\ y) \wedge$ subbagp $(y,\ z)) \rightarrow$ subbagp $(x,\ z)$

THEOREM: not-member-make-negs-fact
$(x \notin y) \rightarrow$ (delete (list (`'rneg`, x), make-negs (y)) = make-negs (y))

THEOREM: make-negs-delete
make-negs (delete $(a,\ x)$) = delete (list (`'rneg`, a), make-negs (x))

THEOREM: make-negs-bagdiff
$\text{make-negs}(\text{bagdiff}(x, y)) = \text{bagdiff}(\text{make-negs}(x), \text{make-negs}(y))$

THEOREM: append-bagdiff-arg1
$\text{subbagp}(a, b) \rightarrow (\text{append}(\text{bagdiff}(b, a), c) = \text{bagdiff}(\text{append}(b, c), a))$

THEOREM: append-bagdiff-arg2
$(\text{bagint}(a, c) = \mathbf{nil})$
$\rightarrow \quad (\text{append}(c, \text{bagdiff}(b, a)) = \text{bagdiff}(\text{append}(c, b), a))$

THEOREM: bagdiff-bagdiff
$\text{bagdiff}(\text{bagdiff}(a, b), c) = \text{bagdiff}(a, \text{append}(b, c))$

THEOREM: member-rneg-make-negs
$(\text{list}(\texttt{'rneg}, x) \in \text{make-negs}(y)) = (x \in y)$

THEOREM: subbagp-make-negs
$\text{subbagp}(\text{make-negs}(x), \text{make-negs}(y)) = \text{subbagp}(x, y)$

THEOREM: rdifference-reduce
$(\text{rdifference}(\text{reduce}(x), y) = \text{rdifference}(x, y))$
$\wedge \quad (\text{rdifference}(x, \text{reduce}(y)) = \text{rdifference}(x, y))$

THEOREM: requal-simple-rplus-x-simple-rneg-x
$\text{requal}(\text{simple-rplus}(x, \text{simple-rneg}(x)), \text{rational}(\texttt{0}, \texttt{1}))$

THEOREM: requal-rplus-x-simple-rneg-x
$\text{requal}(\text{rplus}(x, \text{simple-rneg}(x)), \text{rational}(\texttt{0}, \texttt{1}))$

THEOREM: rplus-x-rneg-x
$\text{rplus}(x, \text{rneg}(x)) = \text{rational}(\texttt{0}, \texttt{1})$

THEOREM: rdifference-x-x
$\text{rdifference}(x, x) = \text{rational}(\texttt{0}, \texttt{1})$

THEOREM: rdifference-rplus-hack
$\text{rdifference}(\text{rplus}(a, b), a) = \text{reduce}(b)$

THEOREM: rdifference-rplus-hack2
$\text{rdifference}(\text{rplus}(b, a), a) = \text{reduce}(b)$

THEOREM: rplus-rdifference-hack
$(\text{rplus}(a, \text{rdifference}(b, a)) = \text{reduce}(b))$
$\wedge \quad (\text{rplus}(\text{rdifference}(b, a), a) = \text{reduce}(b))$

THEOREM: equal-difference-hack1
$(a = \text{rdifference}(b, c)) \rightarrow (\text{rplus}(c, a) = \text{reduce}(b))$

THEOREM: equal-difference-hack2
$(\text{reduce}\,(a) = \text{rdifference}\,(b,\ c)) \rightarrow (\text{rplus}\,(c,\ a) = \text{reduce}\,(b))$

THEOREM: equal-difference-rewrite
$$\begin{array}{l}(\text{fix}\,(b) = \text{fix}\,(d))\\ \rightarrow \quad (((a - b) = (c - d))\\ \qquad = \quad (((b \not< a) \wedge (b \not< c)) \vee (a = c)))\end{array}$$

THEOREM: equal-difference-rewrite2
$$\begin{array}{l}(\text{fix}\,(a) = \text{fix}\,(c))\\ \rightarrow \quad (((a - b) = (c - d))\\ \qquad = \quad (((b \not< a) \wedge (d \not< c)) \vee (\text{fix}\,(b) = \text{fix}\,(d))))\end{array}$$

THEOREM: equal-times-bridge5
$$\begin{array}{l}(((b * a) = (c * d)) \wedge ((x * a) = (c * y)) \wedge (a \not\simeq \mathsf{0}))\\ \rightarrow \quad (((b * y) = (x * d)) = \mathbf{t})\end{array}$$

THEOREM: equal-plus-difference-rewrite
$$\begin{array}{l}(\text{fix}\,(a) = \text{fix}\,(c))\\ \rightarrow \quad ((((a + b) = (c - d)) = ((b \simeq \mathsf{0}) \wedge ((a \simeq \mathsf{0}) \vee (d \simeq \mathsf{0}))))\\ \qquad \wedge \quad (((b + a) = (c - d))\\ \qquad\qquad = \quad ((b \simeq \mathsf{0}) \wedge ((a \simeq \mathsf{0}) \vee (d \simeq \mathsf{0}))))\\ \qquad \wedge \quad (((c - d) = (a + b))\\ \qquad\qquad = \quad ((b \simeq \mathsf{0}) \wedge ((a \simeq \mathsf{0}) \vee (d \simeq \mathsf{0}))))\\ \qquad \wedge \quad (((c - d) = (b + a))\\ \qquad\qquad = \quad ((b \simeq \mathsf{0}) \wedge ((a \simeq \mathsf{0}) \vee (d \simeq \mathsf{0})))))\end{array}$$

THEOREM: lessp-times-bridge1
$$\begin{array}{l}(((c * v) < (\mathit{x1} * \mathit{z1}))\\ \ \wedge \quad ((w * \mathit{x1}) = (c * d))\\ \ \wedge \quad (c \not\simeq \mathsf{0})\\ \ \wedge \quad (d \not\simeq \mathsf{0}))\\ \rightarrow \quad ((v * w) < (d * \mathit{z1}))\end{array}$$

THEOREM: equal-difference
$$\begin{array}{l}((a = (a - b)) = ((a \in \mathbf{N}) \wedge ((a \simeq \mathsf{0}) \vee (b \simeq \mathsf{0}))))\\ \wedge \quad (((a - b) = a) = ((a \in \mathbf{N}) \wedge ((a \simeq \mathsf{0}) \vee (b \simeq \mathsf{0}))))\end{array}$$

THEOREM: requal-simple-rplus-x-x-rewrite
$\text{requal}\,(\text{simple-rplus}\,(x,\ y),\ \text{simple-rplus}\,(x,\ z)) = \text{requal}\,(y,\ z)$

THEOREM: equal-rplus-x-x-rewrite
$(\text{rplus}\,(x,\ y) = \text{rplus}\,(x,\ z)) = (\text{reduce}\,(y) = \text{reduce}\,(z))$

THEOREM: equal-rplus-rdifference-hack
$$\begin{array}{l}(\text{rplus}\,(x,\ y) = \text{rdifference}\,(\text{rplus}\,(x,\ z),\ v))\\ = \quad (\text{reduce}\,(y) = \text{rdifference}\,(z,\ v))\end{array}$$

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

THEOREM: permutation-does-not-effect-rplus
permutation (x, y)
$\rightarrow$ (eval$ ($\mathbf{t}$, rplus-tree (x), a) $=$ eval$ ($\mathbf{t}$, rplus-tree (y), a))

THEOREM: eval$-rplus-tree-zero
eval$ ($\mathbf{t}$, rplus-tree (append (make-negs (x), x)), a) $=$ rational (`0`, `1`)

THEOREM: cancel-zero-fact
(eval$ ($\mathbf{t}$, rplus-tree (z), a) $=$ rational (`0`, `1`))
$\rightarrow$ (eval$ ($\mathbf{t}$, rplus-tree (append (x, z)), a) $=$ eval$ ($\mathbf{t}$, rplus-tree (x), a))

THEOREM: member-car-x-x
(car (x) $\in x$) $=$ listp (x)

THEOREM: member-bagdiff-append
($e \in$ bagdiff (append (x, z), z)) $=$ ($e \in x$)

THEOREM: permutation-transitive
(permutation (x, y) $\wedge$ permutation (y, z)) $\rightarrow$ permutation (x, z)

DEFINITION:
last-cdr (x)
$=$ **if** listp (x) **then** last-cdr (cdr (x))
else x **endif**

THEOREM: bagdiff-x-x
bagdiff (x, x) $=$ last-cdr (x)

THEOREM: bagdiff-append-arg1
bagdiff (append (z, x), z) $= x$

THEOREM: bagdiff-cons-z-z
bagdiff (cons (x, z), z) $=$ cons (x, last-cdr (z))

THEOREM: bagdiff-not-listp
($\neg$ listp (x)) $\rightarrow$ (bagdiff (x, z) $= x$)

THEOREM: bagdiff-car-in
(listp (x) $\wedge$ (car (x) $\in z$))
$\rightarrow$ (bagdiff (x, z) $=$ bagdiff (cdr (x), delete (car (x), z)))

THEOREM: last-cdr-delete
last-cdr (delete $(e,\ x)) =$ last-cdr (x)

THEOREM: member-subbagp2
(subbagp $(x,\ y) \wedge (e \in x)) \rightarrow (e \in y)$

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

THEOREM: subbagp-bagdiff
subbagp $(x,\ y) \rightarrow$ subbagp (bagdiff $(x,\ z)$, bagdiff $(y,\ z))$

THEOREM: permutation-bagdiff
permutation $(x,\ y) \rightarrow$ permutation (bagdiff $(x,\ z)$, bagdiff $(y,\ z))$

THEOREM: permutation-append-arg1-arg2-bridge
permutation (bagdiff (append $(x,\ z),\ z)$, bagdiff (append $(z,\ x),\ z))$

THEOREM: subbagp-transitive-bridge-helper
(subbagp $(y,\ z) \wedge$ subbagp $(z,\ y))$
$\rightarrow$ (((subbagp $(y,\ x) \leftrightarrow$ subbagp $(z,\ x)) = \mathbf{t})$
$\wedge$ ((subbagp $(x,\ y) \leftrightarrow$ subbagp $(x,\ z)) = \mathbf{t}))$

THEOREM: subbagp-transitive-bridge
(subbagp $(y,\ z) \wedge$ subbagp $(z,\ y))$
$\rightarrow$ (((subbagp $(y,\ x) =$ subbagp $(z,\ x)) = \mathbf{t})$
$\wedge$ ((subbagp $(x,\ y) =$ subbagp $(x,\ z)) = \mathbf{t}))$

THEOREM: equal-permutation
(permutation $(y,\ z) \rightarrow$ ((permutation $(x,\ y) =$ permutation $(x,\ z)) = \mathbf{t}))$
$\wedge$ (permutation $(y,\ z) \rightarrow$ ((permutation $(y,\ x) =$ permutation $(z,\ x)) = \mathbf{t}))$

THEOREM: permutation-bagdiff-append-helper
(permutation (bagdiff (append $(z,\ x),\ x),\ y)$
$=$ permutation (bagdiff (append $(x,\ z),\ x),\ y))$
$\wedge$ (permutation (y, bagdiff (append $(z,\ x),\ x))$
$=$ permutation (y, bagdiff (append $(x,\ z),\ x)))$

THEOREM: permutation-bagdiff-append
permutation (x, bagdiff (append $(x,\ z),\ z))$

THEOREM: cancel-zero-fact-bridge
(subbagp $(z,\ x) \wedge$ (eval\$ ($\mathbf{t}$, rplus-tree $(z),\ a) =$ rational $(\mathtt{0},\ \mathtt{1})))$
$\rightarrow$ (eval\$ ($\mathbf{t}$, rplus-tree (bagdiff $(x,\ z)),\ a) =$ eval\$ ($\mathbf{t}$, rplus-tree $(x),\ a))$

THEOREM: rnegs-cancel-list
eval\$ ($\mathbf{t}$, rplus-tree (append (x, make-negs $(x))),\ a) =$ rational $(\mathtt{0},\ \mathtt{1})$

THEOREM: subbagp-append-simplify1
subbagp $(a,\ x) \rightarrow$ subbagp $(a,$ append $(x,\ y))$

THEOREM: subbagp-permutation-equiv
(permutation $(x,\ y) \land$ subbagp $(a,\ x)) \rightarrow$ subbagp $(a,\ y)$

THEOREM: subbagp-append-simplify2
subbagp $(a,\ x) \rightarrow$ subbagp $(a,$ append $(y,\ x))$

THEOREM: subbagp-append-bridge
(subbagp $(x,\ b) \land$ subbagp $(y,\ a)) \rightarrow$ subbagp (append $(x,\ y)$, append $(b,\ a))$

THEOREM: subbagp-append-bridge2
(subbagp $(x,\ b) \land$ subbagp $(y,\ a)) \rightarrow$ subbagp (append $(x,\ y)$, append $(a,\ b))$

THEOREM: cancelling-from-rplus
(subbagp $(z,\ x) \land$ subbagp $(z,\ y) \land$ (bagint (make-negs (z), $x) =$ **nil**))
$\rightarrow$ (eval\$ (**t**, rplus-tree (append (bagdiff $(x,\ z)$, make-negs (bagdiff $(y,\ z)$))), a)
$=$ eval\$ (**t**, rplus-tree (append $(x$, make-negs (y))), a))

THEOREM: eval\$-rplus-tree-rplus-fringe
(car $(x) =$ `'rplus`)
$\rightarrow$ (eval\$ (**t**, rplus-tree (rplus-fringe (x)), $a) =$ eval\$ (**t**, $x,\ a$))

THEOREM: subbagp-x-x
subbagp $(x,\ x)$

THEOREM: rnegs-not-car-split-by-parity
((car $(x) =$ `'rneg`) $\land$ (caadr $(x) \neq$ `'rneg`) $\land$ (cddr $(x) =$ **nil**))
$\rightarrow$ ($x \notin$ car (split-by-parity (y)))

THEOREM: permutation-append-split-by-parity-bridge
permutation (append (car (split-by-parity (x)),
make-negs (cdr (split-by-parity (x)))),
x)

THEOREM: member-cdr-split-by-parity
($e \in$ cdr (split-by-parity (x))) $\rightarrow$ (car $(e) \neq$ `'rneg`)

DEFINITION:
all-rnegs (x)
= **if** listp (x)
then (caar $(x) =$ `'rneg`)
$\land$ (caadar $(x) \neq$ `'rneg`)
$\land$ (cddar $(x) =$ **nil**)
$\land$ all-rnegs (cdr (x))
else t endif

THEOREM: all-rnegs-make-negs-bagint-fact
(all-rnegs (make-negs (bagint (z, w))) $\wedge$ ($v \in w$))
$\rightarrow$ all-rnegs (make-negs (bagint (z, delete (v, w))))

THEOREM: all-rnegs-make-negs-bagint
all-rnegs (make-negs (bagint (car (split-by-parity (x)), cdr (split-by-parity (y)))))

THEOREM: bagint-all-rnegs-car-split-by-parity
all-rnegs (x) $\rightarrow$ (bagint (x, car (split-by-parity (y))) = **nil**)

THEOREM: bagint-make-negs-split-by-parity
bagint (make-negs (bagint (car (split-by-parity (y)), cdr (split-by-parity (y)))),
car (split-by-parity (y)))
= **nil**

THEOREM: commutativity2-of-rtimes
rtimes (x, rtimes (y, z)) = rtimes (y, rtimes (x, z))

EVENT: Enable eval\$-rplus-tree-rplus-fringe; name this event 'eval\$-rplus-tree-rplus-fringe-off'.

EVENT: Enable cancelling-from-rplus; name this event 'cancelling-from-rplus-off'.

EVENT: Enable rnegs-cancel-list; name this event 'rnegs-cancel-list-off'.

EVENT: Enable cancel-zero-fact-bridge; name this event 'cancel-zero-fact-bridge-off'.

EVENT: Enable cancel-zero-fact; name this event 'cancel-zero-fact-off'.

EVENT: Enable eval\$-rplus-tree-zero; name this event 'eval\$-rplus-tree-zero-off'.

EVENT: Enable permutation-does-not-effect-rplus; name this event 'permutation-does-not-effect-rplus-off'.

EVENT: Enable eval\$-rplus-tree-delete; name this event 'eval\$-rplus-tree-delete-off'.

EVENT: Enable rplus-eval\$-rplus-tree; name this event 'rplus-eval\$-rplus-tree-off'.

EVENT: Enable reduce-eval$-rplus-tree; name this event 'reduce-eval$-rplus-tree-off'.

EVENT: Enable bagint-make-negs-split-by-parity; name this event 'bagint-make-negs-split-by-parity-off'.

EVENT: Enable bagint-all-rnegs-car-split-by-parity; name this event 'bagint-all-rnegs-car-split-by-parity-off'.

EVENT: Enable all-rnegs-make-negs-bagint; name this event 'all-rnegs-make-negs-bagint-off'.

EVENT: Enable all-rnegs-make-negs-bagint-fact; name this event 'all-rnegs-make-negs-bagint-fact-off'.

EVENT: Enable member-cdr-split-by-parity; name this event 'member-cdr-split-by-parity-off'.

EVENT: Enable permutation-append-split-by-parity-bridge; name this event 'permutation-append-split-by-parity-bridge-off'.

EVENT: Enable rnegs-not-car-split-by-parity; name this event 'rnegs-not-car-split-by-parity-off'.

EVENT: Enable subbagp-x-x; name this event 'subbagp-x-x-off'.

EVENT: Enable subbagp-append-bridge2; name this event 'subbagp-append-bridge2-off'.

EVENT: Enable subbagp-append-bridge; name this event 'subbagp-append-bridge-off'.

EVENT: Enable subbagp-append-simplify2; name this event 'subbagp-append-simplify2-off'.

EVENT: Enable subbagp-permutation-equiv; name this event 'subbagp-permutation-equiv-off'.

EVENT: Enable subbagp-append-simplify1; name this event ‘subbagp-append-simplify1-off’.

EVENT: Enable permutation-bagdiff-append; name this event ‘permutation-bagdiff-append-off’.

EVENT: Enable permutation-bagdiff-append-helper; name this event ‘permutation-bagdiff-append-helper-off’.

EVENT: Enable equal-permutation; name this event ‘equal-permutation-off’.

EVENT: Enable subbagp-transitive-bridge; name this event ‘subbagp-transitive-bridge-off’.

EVENT: Enable subbagp-transitive-bridge-helper; name this event ‘subbagp-transitive-bridge-helper-off’.

EVENT: Enable permutation-append-arg1-arg2-bridge; name this event ‘permutation-append-arg1-arg2-bridge-off’.

EVENT: Enable permutation-bagdiff; name this event ‘permutation-bagdiff-off’.

EVENT: Enable subbagp-bagdiff; name this event ‘subbagp-bagdiff-off’.

EVENT: Enable member-subbagp-delete; name this event ‘member-subbagp-delete-off’.

EVENT: Enable member-subbagp2; name this event ‘member-subbagp2-off’.

EVENT: Enable last-cdr-delete; name this event ‘last-cdr-delete-off’.

EVENT: Enable bagdiff-car-in; name this event ‘bagdiff-car-in-off’.

EVENT: Enable bagdiff-not-listp; name this event ‘bagdiff-not-listp-off’.

EVENT: Enable bagdiff-cons-z-z; name this event ‘bagdiff-cons-z-z-off’.

EVENT: Enable bagdiff-append-arg1; name this event ‘bagdiff-append-arg1-off’.

EVENT: Enable bagdiff-x-x; name this event ‘bagdiff-x-x-off’.

EVENT: Enable permutation-transitive; name this event ‘permutation-transitive-off’.

EVENT: Enable member-bagdiff-append; name this event ‘member-bagdiff-append-off’.

EVENT: Enable member-car-x-x; name this event ‘member-car-x-x-off’.

EVENT: Enable equal-rplus-rdifference-hack; name this event ‘equal-rplus-rdifference-hack-off’.

EVENT: Enable equal-rplus-x-x-rewrite; name this event ‘equal-rplus-x-x-rewrite-off’.

EVENT: Enable requal-simple-rplus-x-x-rewrite; name this event ‘requal-simple-rplus-x-x-rewrite-off’.

EVENT: Enable equal-difference; name this event ‘equal-difference-off’.

EVENT: Enable lessp-times-bridge1; name this event ‘lessp-times-bridge1-off’.

EVENT: Enable equal-plus-difference-rewrite; name this event ‘equal-plus-difference-rewrite-off’.

EVENT: Enable equal-times-bridge5; name this event ‘equal-times-bridge5-off’.

EVENT: Enable equal-difference-rewrite2; name this event ‘equal-difference-rewrite2-off’.

EVENT: Enable equal-difference-rewrite; name this event ‘equal-difference-rewrite-off’.

EVENT: Enable equal-difference-hack2; name this event ‘equal-difference-hack2-

off'.

Event: Enable equal-difference-hack1; name this event 'equal-difference-hack1-off'.

Event: Enable rplus-rdifference-hack; name this event 'rplus-rdifference-hack-off'.

Event: Enable rdifference-rplus-hack2; name this event 'rdifference-rplus-hack2-off'.

Event: Enable rdifference-rplus-hack; name this event 'rdifference-rplus-hack-off'.

Event: Enable rdifference-x-x; name this event 'rdifference-x-x-off'.

Event: Enable rplus-x-rneg-x; name this event 'rplus-x-rneg-x-off'.

Event: Enable requal-rplus-x-simple-rneg-x; name this event 'requal-rplus-x-simple-rneg-x-off'.

Event: Enable requal-simple-rplus-x-simple-rneg-x; name this event 'requal-simple-rplus-x-simple-rneg-x-off'.

Event: Enable rdifference-reduce; name this event 'rdifference-reduce-off'.

Event: Enable subbagp-make-negs; name this event 'subbagp-make-negs-off'.

Event: Enable member-rneg-make-negs; name this event 'member-rneg-make-negs-off'.

Event: Enable bagdiff-bagdiff; name this event 'bagdiff-bagdiff-off'.

Event: Enable append-bagdiff-arg2; name this event 'append-bagdiff-arg2-off'.

Event: Enable append-bagdiff-arg1; name this event 'append-bagdiff-arg1-off'.

EVENT: Enable make-negs-bagdiff; name this event 'make-negs-bagdiff-off'.

EVENT: Enable make-negs-delete; name this event 'make-negs-delete-off'.

EVENT: Enable not-member-make-negs-fact; name this event 'not-member-make-negs-fact-off'.

EVENT: Enable subbagp-transitive; name this event 'subbagp-transitive-off'.

EVENT: Enable subbagp-necc; name this event 'subbagp-necc-off'.

EVENT: Enable not-subbagp-not-permutation; name this event 'not-subbagp-not-permutation-off'.

EVENT: Enable permutation-a-b; name this event 'permutation-a-b-off'.

EVENT: Enable subbagp-permutation; name this event 'subbagp-permutation-off'.

EVENT: Enable subbagp-delete-car2; name this event 'subbagp-delete-car2-off'.

EVENT: Enable subbagp-delete-car; name this event 'subbagp-delete-car-off'.

EVENT: Enable subbagp-delete-same-means; name this event 'subbagp-delete-same-means-off'.

EVENT: Enable subbagp-delete-same; name this event 'subbagp-delete-same-off'.

EVENT: Enable subbagp-append; name this event 'subbagp-append-off'.

EVENT: Enable occurrences-append; name this event 'occurrences-append-off'.

EVENT: Enable occurrences-delete2; name this event 'occurrences-delete2-off'.

EVENT: Enable member-subbagp; name this event 'member-subbagp-off'.

EVENT: Enable delete-append; name this event 'delete-append-off'.

EVENT: Enable member-append; name this event 'member-append-off'.

EVENT: Enable eval$-reduce; name this event 'eval$-reduce-off'.

EVENT: Enable eval$-rplus; name this event 'eval$-rplus-off'.

EVENT: Enable rplus-rzerop; name this event 'rplus-rzerop-off'.

EVENT: Enable rplus-rzerop-bridge2; name this event 'rplus-rzerop-bridge2-off'.

EVENT: Enable rplus-rzerop-bridge; name this event 'rplus-rzerop-bridge-off'.

DEFINITION:
cancel-rplus (x)
= **if** car (x) = `'rplus`
  then if listp (bagint (car (split-by-parity (rplus-fringe (x))),
                cdr (split-by-parity (rplus-fringe (x)))))
    then rplus-tree (append (bagdiff (car (split-by-parity (rplus-fringe (x))),
                bagint (car (split-by-parity (rplus-fringe (x))),
                  cdr (split-by-parity (rplus-fringe (x))))),
              make-negs (bagdiff (cdr (split-by-parity (rplus-fringe (x))),
                bagint (car (split-by-parity (rplus-fringe (x))),
                  cdr (split-by-parity (rplus-fringe (x))))))))
    else x **endif**
  else x **endif**

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

EVENT: Enable correctness-of-cancel-rplus; name this event 'correctness-of-cancel-rplus-off'.

EVENT: Enable cancel-rplus; name this event 'cancel-rplus-off'.

EVENT: Let us define the theory *r2* to consist of the following events: rtimes-rneg-arg2, rtimes-rneg-arg1, rtimes-rplus-arg2, rtimes-rplus-arg1, associativity-of-rtimes, rzerop, commutativity-of-rtimes, rdifference-rdifference-arg2, rdifference-rdifference-arg1, rneg-rplus, rplus-reduce-arg2-rewrite, rplus-reduce-arg1-rewrite, reduce-rneg, reduce-rmagnitude, reduce-rquotient, reduce-difference, reduce-rtimes, reduce-rplus, commutativity2-of-rplus, associativity-of-rplus, equal-requal-rewrite, transitivity-of-requal, fix-rational-of-rationalp, nrational-rplus-arg2, nrational-rplus-arg1, rationalp-means, means-rationalp, rneg-rneg, rneg-reduce, reduce-0, numberp-numerator-reduce, reduce-nrationalp, rplus-reduce-arg2, rplus-reduce-arg1, requal-x-x, rplus-requal-arg1, commutativity-of-rplus, reduce-reduce, requal-reduce2, requal-reduce1, commutativity-of-requal, rational-generalization, fix-rational-rmagnitude, fix-rational-rquotient, fix-rational-rdifference, fix-rational-rneg, fix-rational-rtimes, fix-rational-fix-rational, fix-rational-rplus, fix-rational-reduce, rationalp-rmagnitude, rationalp-rquotient, rationalp-rdifference, rationalp-rneg, rationalp-rtimes, rationalp-fix-rational, rationalp-rplus, rationalp-reduce, commutativity2-of-rtimes, rdifference, correctness-of-cancel-rplus.

EVENT: Make the library `"r2"`.

Index