

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

DEFINITION:

delete(x , l)
= **if** listp(l)
 then if $x = \text{car}(l)$ **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 $x = \text{car}(l)$ **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 $\text{car}(x) \in y$ **then** $\text{subbagp}(\text{cdr}(x), \text{delete}(\text{car}(x), y))$
else f endif
else t endif

THEOREM: listp-delete

$\text{listp}(\text{delete}(x, l))$
 $=$ **if** $\text{listp}(l)$ **then** $(x \neq \text{car}(l)) \vee \text{listp}(\text{cdr}(l))$
else f endif

THEOREM: delete-non-member

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

THEOREM: delete-delete

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

THEOREM: equal-occurrences-zero

$(\text{occurrences}(x, l) = 0) = (x \notin l)$

THEOREM: occurrences-in-a-set

$\text{setp}(l)$
 $\rightarrow$ $(\text{occurrences}(x, l)$
 $=$ **if** $x \in l$ **then** 1
else 0 endif)

THEOREM: occurrences-cons

$\text{occurrences}(x, \text{cons}(y, l))$
 $=$ **if** $x = y$ **then** $1 + \text{occurrences}(x, l)$
else $\text{occurrences}(x, l)$ **endif**

THEOREM: occurrences-delete

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

THEOREM: occurrences-bagdiff

$\text{occurrences}(x, \text{bagdiff}(a, b)) = (\text{occurrences}(x, a) - \text{occurrences}(x, b))$

THEOREM: occurrences-bagint

$\text{occurrences}(x, \text{bagint}(a, b))$
 $=$ **if** $\text{occurrences}(x, a) < \text{occurrences}(x, b)$ **then** $\text{occurrences}(x, a)$
else $\text{occurrences}(x, b)$ **endif**

THEOREM: member-non-list

$(\neg \text{listp}(l)) \rightarrow (x \notin l)$

THEOREM: member-cons

$$(x \in \text{cons}(y, l)) = ((x = y) \vee (x \in l))$$

THEOREM: member-delete-implies-membership

$$(x \in \text{delete}(y, l)) \rightarrow (x \in l)$$

THEOREM: member-delete

$$\begin{aligned} & (x \in \text{delete}(y, l)) \\ = & \text{ if } x \in l \\ & \text{ then if } x = y \text{ then } 1 < \text{occurrences}(x, l) \\ & \text{ else t endif} \\ & \text{ else f endif} \end{aligned}$$

THEOREM: permutation-implies-membership-equivalence

$$\text{permutation}(a, b) \rightarrow ((x \in a) = (x \in b))$$

THEOREM: member-bagdiff

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

THEOREM: member-bagint

$$(x \in \text{bagint}(a, b)) = ((x \in a) \wedge (x \in b))$$

THEOREM: not-permutation

$$(\text{listp}(b) \wedge (\text{car}(b) \notin a)) \rightarrow (\neg \text{permutation}(a, b))$$

THEOREM: permutation-right-cons1

$$\begin{aligned} & (\text{listp}(b) \wedge (\text{car}(b) \in a)) \\ \rightarrow & (\text{permutation}(a, b) = \text{permutation}(\text{delete}(\text{car}(b), a), \text{cdr}(b))) \end{aligned}$$

THEOREM: permutation-right-cons

$$\begin{aligned} & \text{permutation}(a, \text{cons}(x, b)) \\ = & \text{ if } x \in a \text{ then } \text{permutation}(\text{delete}(x, a), b) \\ & \text{ else f endif} \end{aligned}$$

THEOREM: commutativity-of-permutation

$$\text{permutation}(b, a) = \text{permutation}(a, b)$$

THEOREM: permutation-reflexivity

$$\text{permutation}(l, l)$$

THEOREM: permutation-delete-delete

$$\begin{aligned} & (\text{permutation}(a, b) \wedge (x \in a) \wedge (x \in b)) \\ \rightarrow & \text{permutation}(\text{delete}(x, a), \text{delete}(x, b)) \end{aligned}$$

DEFINITION:

transitivity-of-permutation-induction (a, b, c)

= **if** listp (a)
 then transitivity-of-permutation-induction (cdr (a) ,
 delete (car (a) , b),
 delete (car (a) , c))
 else 0 endif

THEOREM: transitivity-of-permutation-base-case

$((a \simeq \mathbf{nil}) \wedge \text{permutation}(a, b) \wedge \text{permutation}(b, c)) \rightarrow \text{permutation}(a, c)$

THEOREM: transitivity-of-permutation-induction-step

(listp (a))
 $\wedge$ ((permutation (cdr (a) , delete (car (a) , b))
 $\wedge$ permutation (delete (car (a) , b), delete (car (a) , c)))
 $\rightarrow$ permutation (cdr (a) , delete (car (a) , c)))
 $\wedge$ permutation (a, b)
 $\wedge$ permutation (b, c)
 $\rightarrow$ permutation (a, c)

THEOREM: transitivity-of-permutation

$(\text{permutation}(a, b) \wedge \text{permutation}(b, c)) \rightarrow \text{permutation}(a, c)$

THEOREM: setp-delete

setp $(a) \rightarrow \text{setp}(\text{delete}(x, a))$

DEFINITION:

setp-permutation-induction (a, b)

= **if** listp (a) **then** setp-permutation-induction (cdr (a) , delete (car (a) , b))
 else 0 endif

THEOREM: setp-permutation-base-case

$(\neg \text{listp}(a)) \rightarrow \text{setp}(a)$

THEOREM: setp-permutation-induction-step

(listp (a))
 $\wedge$ setp (b)
 $\wedge$ permutation (a, b)
 $\wedge$ ((setp (delete (car (a) , b)) $\wedge$ permutation (cdr (a) , delete (car (a) , b)))
 $\rightarrow$ setp (cdr (a))))
 $\rightarrow$ setp (a)

THEOREM: setp-permutation

$(\text{setp}(b) \wedge \text{permutation}(a, b)) \rightarrow \text{setp}(a)$

THEOREM: subbagp-delete

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

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

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

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

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

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
 $(x + (1 + y))$
 $= \text{ if } y \in \mathbf{N} \text{ then } 1 + (x + y)$
 $\text{ else } 1 + x \text{ endif}$

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

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

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

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

THEOREM: equal-plus-1

$$\begin{aligned} & ((a + b) = 1) \\ = & (((a \simeq 0) \wedge (b = 1)) \vee ((a = 1) \wedge (b \simeq 0))) \end{aligned}$$

THEOREM: plus-cancellation

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

THEOREM: plus-difference-arg1-casesplit

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

THEOREM: plus-difference-arg1

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

THEOREM: plus-difference-arg2-casesplit

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

THEOREM: plus-difference-arg2

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

THEOREM: plus-difference-difference-casesplit

$$\begin{aligned} & ((a - b) + (c - d)) \\ = & \text{if if } a < b \text{ then f} \\ & \text{else t endif} \\ & \text{then if if } c < d \text{ then f} \\ & \text{else t endif then } (a + c) - (b + d) \\ & \text{else } a - b \text{ endif} \\ & \text{elseif if } c < d \text{ then f} \\ & \text{else t endif then } c - d \\ & \text{else 0 endif} \end{aligned}$$

THEOREM: plus-difference-difference

$$\begin{aligned} & ((b < a) \wedge (d < c)) \\ \rightarrow & (((a - b) + (c - d)) = ((a + c) - (b + d))) \end{aligned}$$

THEOREM: difference-elim

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

THEOREM: difference-leq-arg1

$$\begin{aligned} & \text{if } b < a \text{ then f} \\ & \text{else t endif} \\ \rightarrow & ((a - b) = 0) \end{aligned}$$

THEOREM: difference-x-x

$$(x - x) = 0$$

THEOREM: equal-difference-0

$$((0 = (x - y)) = (y \not< x)) \wedge (((x - y) = 0) = (y \not< x))$$

THEOREM: difference-plus

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

THEOREM: difference-plus-cancellation1

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

THEOREM: difference-plus-cancellation2

$$((a + c) - (b + c)) = (a - b)$$

THEOREM: difference-cancellation-0

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

THEOREM: difference-cancellation-1

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

THEOREM: difference-1

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

THEOREM: difference-2

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

THEOREM: difference-add1-plus-cancellation

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

THEOREM: difference-sub1-arg2

$$\begin{aligned} & (a - (b - 1)) \\ & = \text{if if } 1 < b \text{ then f} \\ & \quad \text{else t endif then fix}(a) \\ & \quad \text{elseif } a < b \text{ then } 0 \\ & \quad \text{else } 1 + (a - b) \text{ endif} \end{aligned}$$

THEOREM: difference-difference-arg1

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

THEOREM: difference-difference-arg2

$$\begin{aligned} & (a - (b - c)) \\ & = \text{if if } b < c \text{ then f} \\ & \quad \text{else t endif then } (a + c) - b \\ & \quad \text{else fix}(a) \text{ endif} \end{aligned}$$

THEOREM: difference-difference-difference

```
(if a < b then f
  else t endif
  ∧ if c < d then f
    else t endif)
→ (((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)
= if listp(x) ∧ (car(x) = 'plus)
  then append(plus-fringe(cadr(x)), plus-fringe(caddr(x)))
  else list(x) endif
```

DEFINITION:

```
plus-tree(l)
= if l ≃ nil then ''0
  elseif cdr(l) ≃ nil then list('fix, car(l))
  elseif caddr(l) ≃ nil then list('plus, car(l), cadr(l))
  else list('plus, car(l), plus-tree(cdr(l))) endif
```

THEOREM: numberp-eval\$-plus

```
(listp(x) ∧ (car(x) = 'plus)) → (eval$(t, x, a) ∈ N)
```

THEOREM: numberp-eval\$-plus-tree

```
eval$(t, plus-tree(l), a) ∈ N
```

THEOREM: member-implies-plus-tree-greatereqp

```
(x ∈ y) → (eval$(t, plus-tree(y), a) ⋈ eval$(t, x, a))
```

THEOREM: plus-tree-delete

```
eval$(t, plus-tree(delete(x, y)), a)
= if x ∈ y then eval$(t, plus-tree(y), a) - eval$(t, x, a)
  else eval$(t, plus-tree(y), a) endif
```

THEOREM: subbagp-implies-plus-tree-greatereqp

```
subbagp(x, y) → (eval$(t, plus-tree(y), a) ⋈ eval$(t, plus-tree(x), a))
```

THEOREM: plus-tree-bagdiff

```
subbagp(x, y)
→ (eval$(t, plus-tree(bagdiff(y, x)), a)
   = (eval$(t, plus-tree(y), a) - eval$(t, plus-tree(x), a)))
```

THEOREM: numberp-eval\$-bridge

```
(eval$(t, z, a) = eval$(t, plus-tree(x), a)) → (eval$(t, z, a) ∈ N)
```


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

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

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

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

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

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

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

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

```

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

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

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

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

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

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

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

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

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

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

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

THEOREM: diff-x-x

$$(x - x) = 0$$

THEOREM: lessp-difference-cancellation

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

THEOREM: lessp-difference

$$((x \neq 0) \wedge (y \neq 0)) \rightarrow (((x - y) < x) = \mathbf{t})$$

DEFINITION:

cancel-lessp-plus(x)

$$\begin{aligned} = & \text{if listp}(x) \wedge (\text{car}(x) = \text{'lessp'}) \\ & \text{then if listp}(\text{cadr}(x)) \\ & \quad \wedge (\text{caadr}(x) = \text{'plus'}) \\ & \quad \wedge \text{listp}(\text{caddr}(x)) \\ & \quad \wedge (\text{caaddr}(x) = \text{'plus'}) \\ & \quad \text{then list}(\text{'lessp'}, \\ & \quad \quad \text{plus-tree}(\text{bagdiff}(\text{plus-fringe}(\text{cadr}(x)), \\ & \quad \quad \quad \text{bagint}(\text{plus-fringe}(\text{cadr}(x)), \\ & \quad \quad \quad \text{plus-fringe}(\text{caddr}(x))))) \\ & \quad \quad \text{plus-tree}(\text{bagdiff}(\text{plus-fringe}(\text{caddr}(x)), \\ & \quad \quad \quad \text{bagint}(\text{plus-fringe}(\text{cadr}(x)), \\ & \quad \quad \quad \text{plus-fringe}(\text{caddr}(x))))) \\ & \quad \text{elseif listp}(\text{cadr}(x)) \\ & \quad \quad \wedge (\text{caadr}(x) = \text{'plus'}) \\ & \quad \quad \wedge (\text{caddr}(x) \in \text{plus-fringe}(\text{cadr}(x))) \\ & \quad \quad \text{then list}(\text{'quote', f}) \\ & \quad \quad \text{elseif listp}(\text{caddr}(x)) \\ & \quad \quad \quad \wedge (\text{caaddr}(x) = \text{'plus'}) \\ & \quad \quad \quad \wedge (\text{cadr}(x) \in \text{plus-fringe}(\text{caddr}(x))) \\ & \quad \quad \text{then list}(\text{'not',} \\ & \quad \quad \quad \text{list}(\text{'zerop',} \\ & \quad \quad \quad \text{plus-tree}(\text{delete}(\text{cadr}(x), \text{plus-fringe}(\text{caddr}(x))))) \\ & \quad \quad \text{else } x \text{ endif} \\ & \text{else } x \text{ endif} \end{aligned}$$

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:

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

THEOREM: times-zero

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

THEOREM: times-add1

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

THEOREM: times-sub1

$$(b \not\simeq 0) \rightarrow ((a * (b - 1)) = ((a * b) - a))$$

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

$$((x * (y + z)) = ((x * y) + (x * z))) \\ \wedge (((x + y) * z) = ((x * z) + (y * z)))$$

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

$$\begin{aligned} &(((a - b) * c) = ((a * c) - (b * c))) \\ \wedge \quad &((a * (b - c)) = ((a * b) - (a * c))) \end{aligned}$$

THEOREM: times-quotient-proof

$$((x \neq 0) \wedge ((y \mathbf{mod} x) = 0)) \rightarrow (((y \div x) * x) = \mathbf{fix}(y))$$

THEOREM: times-quotient

$$\begin{aligned} &((y \neq 0) \wedge ((x \mathbf{mod} y) = 0)) \\ \rightarrow \quad &(((x \div y) * y) = \mathbf{fix}(x)) \wedge ((y * (x \div y)) = \mathbf{fix}(x)) \end{aligned}$$

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

$$(1 * x) = \mathbf{fix}(x)$$

THEOREM: lessp-times1-proof

$$((a < b) \wedge (c \neq 0)) \rightarrow ((a < (b * c)) = \mathbf{t})$$

THEOREM: lessp-times1

$$\begin{aligned} &((a < b) \wedge (c \neq 0)) \\ \rightarrow \quad &(((a < (b * c)) = \mathbf{t}) \wedge ((a < (c * b)) = \mathbf{t})) \end{aligned}$$

THEOREM: lessp-times2-proof

$$\begin{aligned} &(\mathbf{if} \ b < a \ \mathbf{then} \ \mathbf{f} \\ &\quad \mathbf{else} \ \mathbf{t} \ \mathbf{endif} \\ &\wedge \quad (c \neq 0)) \\ \rightarrow \quad &(((b * c) < a) = \mathbf{f}) \end{aligned}$$

THEOREM: lessp-times2

$$\begin{aligned} &(\mathbf{if} \ b < a \ \mathbf{then} \ \mathbf{f} \\ &\quad \mathbf{else} \ \mathbf{t} \ \mathbf{endif} \\ &\wedge \quad (c \neq 0)) \\ \rightarrow \quad &(((b * c) < a) = \mathbf{f}) \wedge (((c * b) < a) = \mathbf{f}) \end{aligned}$$

THEOREM: lessp-times3-proof1

$$((a \neq 0) \wedge (1 < b)) \rightarrow (a < (a * b))$$

THEOREM: lessp-times3-proof2

$$(a < (a * b)) \rightarrow ((a \not\succeq 0) \wedge (1 < b))$$

THEOREM: lessp-times3

$$\begin{aligned} & ((a < (a * b)) = ((a \not\succeq 0) \wedge (1 < b))) \\ \wedge \quad & ((a < (b * a)) = ((a \not\succeq 0) \wedge (1 < b))) \end{aligned}$$

THEOREM: lessp-times-cancellation-proof

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

THEOREM: lessp-times-cancellation1

$$\begin{aligned} & (((x * z) < (y * z)) = ((z \not\succeq 0) \wedge (x < y))) \\ \wedge \quad & (((z * x) < (y * z)) = ((z \not\succeq 0) \wedge (x < y))) \\ \wedge \quad & (((x * z) < (z * y)) = ((z \not\succeq 0) \wedge (x < y))) \\ \wedge \quad & (((z * x) < (z * y)) = ((z \not\succeq 0) \wedge (x < y))) \end{aligned}$$

THEOREM: lessp-times-cancellation2

$$\begin{aligned} & (((b * a * c) < (a * d)) = ((a \not\succeq 0) \wedge ((b * c) < d))) \\ \wedge \quad & (((((a * c) * b) < (a * d)) = ((a \not\succeq 0) \wedge ((b * c) < d))) \\ & \quad = ((a \not\succeq 0) \wedge ((b * c) < d))) \\ \wedge \quad & (((((b * a * c) < (d * a)) = ((a \not\succeq 0) \wedge ((b * c) < d))) \\ \wedge \quad & (((((a * c) * b) < (d * a)) = ((a \not\succeq 0) \wedge ((b * c) < d))) \\ & \quad = ((a \not\succeq 0) \wedge ((b * c) < d))) \\ \wedge \quad & (((((b * c * a) < (a * d)) = ((a \not\succeq 0) \wedge ((b * c) < d))) \\ \wedge \quad & (((((c * a) * b) < (a * d)) = ((a \not\succeq 0) \wedge ((b * c) < d))) \\ & \quad = ((a \not\succeq 0) \wedge ((b * c) < d))) \\ \wedge \quad & (((((b * c * a) < (d * a)) = ((a \not\succeq 0) \wedge ((b * c) < d))) \\ \wedge \quad & (((((c * a) * b) < (d * a)) = ((a \not\succeq 0) \wedge ((b * c) < d))) \\ & \quad = ((a \not\succeq 0) \wedge ((b * c) < d))) \end{aligned}$$

THEOREM: lessp-plus-times-proof

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

THEOREM: lessp-plus-times1

$$\begin{aligned} & (((a + (b * c)) < b) = ((a < b) \wedge (c \simeq 0))) \\ \wedge \quad & (((a + (c * b)) < b) = ((a < b) \wedge (c \simeq 0))) \\ \wedge \quad & (((c * b) + a) < b) = ((a < b) \wedge (c \simeq 0))) \\ \wedge \quad & (((b * c) + a) < b) = ((a < b) \wedge (c \simeq 0))) \end{aligned}$$

THEOREM: lessp-plus-times2

$$\begin{aligned} & ((a \not\succeq 0) \wedge (x < a)) \\ \rightarrow \quad & (((x + (a * b)) < (a * c)) = (b < c)) \\ \wedge \quad & (((x + (b * a)) < (a * c)) = (b < c)) \\ \wedge \quad & (((x + (a * b)) < (c * a)) = (b < c)) \\ \wedge \quad & (((x + (b * a)) < (c * a)) = (b < c)) \end{aligned}$$

$$\begin{aligned}
& \wedge \quad (((a * b) + x) < (a * c)) = (b < c) \\
& \wedge \quad (((b * a) + x) < (a * c)) = (b < c) \\
& \wedge \quad (((a * b) + x) < (c * a)) = (b < c) \\
& \wedge \quad (((b * a) + x) < (c * a)) = (b < c)
\end{aligned}$$

THEOREM: lessp-1-times

$$\begin{aligned}
& (1 < (a * b)) \\
& = \quad (\neg ((a \simeq 0) \vee (b \simeq 0) \vee ((a = 1) \wedge (b = 1))))
\end{aligned}$$

THEOREM: equal-sub1-times-0

$$\begin{aligned}
& (((a * b) - 1) = 0) \\
& = \quad ((a \simeq 0) \vee (b \simeq 0) \vee ((a = 1) \wedge (b = 1)))
\end{aligned}$$

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

$$((x \mathbf{mod} y) < y) = (y \not\simeq 0)$$

THEOREM: remainder-noop

$$(a < b) \rightarrow ((a \mathbf{mod} b) = \text{fix}(a))$$

THEOREM: remainder-of-non-number

$$(a \notin \mathbf{N}) \rightarrow ((a \mathbf{mod} n) = (0 \mathbf{mod} n))$$

THEOREM: remainder-zero

$$(x \simeq 0) \rightarrow ((y \mathbf{mod} x) = \text{fix}(y))$$

THEOREM: plus-remainder-times-quotient

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

THEOREM: remainder-quotient-elim

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

THEOREM: remainder-sub1

$$\begin{aligned}
& ((a \not\simeq 0) \wedge (b \not\simeq 0)) \\
& \rightarrow \quad (((a - 1) \mathbf{mod} b) \\
& \quad = \quad \mathbf{if} \ (a \mathbf{mod} b) = 0 \ \mathbf{then} \ b - 1 \\
& \quad \quad \mathbf{else} \ (a \mathbf{mod} b) - 1 \ \mathbf{endif})
\end{aligned}$$

THEOREM: remainder-add1

$$((a \bmod b) = 0) \rightarrow (((1 + a) \bmod b) = (1 \bmod b))$$

THEOREM: remainder-plus-proof

$$((b \bmod c) = 0) \rightarrow (((a + b) \bmod c) = (a \bmod c))$$

THEOREM: remainder-plus

$$\begin{aligned} & ((a \bmod c) = 0) \\ \rightarrow & \quad (((a + b) \bmod c) = (b \bmod c)) \\ & \quad \wedge \quad (((b + a) \bmod c) = (b \bmod c)) \\ & \quad \wedge \quad (((x + y + a) \bmod c) = ((x + y) \bmod c)) \end{aligned}$$

THEOREM: equal-remainder-plus-0-proof

$$((a \bmod c) = 0) \rightarrow (((a + b) \bmod c) = 0) = ((b \bmod c) = 0))$$

THEOREM: equal-remainder-plus-0

$$\begin{aligned} & ((a \bmod c) = 0) \\ \rightarrow & \quad (((a + b) \bmod c) = 0) = ((b \bmod c) = 0)) \\ & \quad \wedge \quad (((b + a) \bmod c) = 0) = ((b \bmod c) = 0)) \\ & \quad \wedge \quad (((x + y + a) \bmod c) = 0) \\ & \quad \quad = \quad (((x + y) \bmod c) = 0)) \end{aligned}$$

THEOREM: equal-remainder-plus-remainder-proof

$$(a < c) \rightarrow (((a + b) \bmod c) = (b \bmod c)) = (a \simeq 0))$$

THEOREM: equal-remainder-plus-remainder

$$\begin{aligned} & (a < c) \\ \rightarrow & \quad (((a + b) \bmod c) = (b \bmod c)) = (a \simeq 0)) \\ & \quad \wedge \quad (((b + a) \bmod c) = (b \bmod c)) = (a \simeq 0)) \end{aligned}$$

THEOREM: remainder-times1-proof

$$((b \bmod c) = 0) \rightarrow (((a * b) \bmod c) = 0)$$

THEOREM: remainder-times1

$$\begin{aligned} & ((b \bmod c) = 0) \\ \rightarrow & \quad (((a * b) \bmod c) = 0) \wedge (((b * a) \bmod c) = 0)) \end{aligned}$$

THEOREM: remainder-times1-instance-proof

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

THEOREM: remainder-times1-instance

$$(((x * y) \bmod y) = 0) \wedge (((x * y) \bmod x) = 0)$$

THEOREM: remainder-times-times-proof

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

THEOREM: remainder-times-times

$$\begin{aligned} & (((x * y) \bmod (x * z)) = (x * (y \bmod z))) \\ \wedge & (((x * z) \bmod (y * z)) = ((x \bmod y) * z)) \end{aligned}$$

THEOREM: remainder-times2-proof

$$((a \bmod z) = 0) \rightarrow ((a \bmod (z * y)) = (z * ((a \div z) \bmod y)))$$

THEOREM: remainder-times2

$$\begin{aligned} & ((a \bmod z) = 0) \\ \rightarrow & (((a \bmod (y * z)) = (z * ((a \div z) \bmod y))) \\ & \wedge ((a \bmod (z * y)) = (z * ((a \div z) \bmod y)))) \end{aligned}$$

THEOREM: remainder-times2-instance

$$\begin{aligned} & (((x * y) \bmod (x * z)) = (x * (y \bmod z))) \\ \wedge & (((x * z) \bmod (y * z)) = ((x \bmod y) * z)) \end{aligned}$$

THEOREM: remainder-difference1

$$\begin{aligned} & ((a \bmod c) = (b \bmod c)) \\ \rightarrow & (((a - b) \bmod c) = ((a \bmod c) - (b \bmod c))) \end{aligned}$$

DEFINITION:

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

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

THEOREM: remainder-difference2

$$\begin{aligned} & (((a \bmod c) = 0) \wedge ((b \bmod c) \neq 0)) \\ \rightarrow & (((a - b) \bmod c) \\ & = \text{ if } b < a \text{ then } c - (b \bmod c) \\ & \text{ else } 0 \text{ endif}) \end{aligned}$$

THEOREM: remainder-difference3

$$\begin{aligned} & (((b \bmod c) = 0) \wedge ((a \bmod c) \neq 0)) \\ \rightarrow & (((a - b) \bmod c) \\ & = \text{ if } b < a \text{ then } a \bmod c \\ & \text{ else } 0 \text{ endif}) \end{aligned}$$

THEOREM: equal-remainder-difference-0

$$\begin{aligned} & (((a - b) \bmod c) = 0) \\ = & \text{ if if } a < b \text{ then f} \\ & \text{ else t endif then } (a \bmod c) = (b \bmod c) \\ & \text{ else t endif} \end{aligned}$$

THEOREM: lessp-plus-fact

$$\begin{aligned} & (((b \bmod x) = 0) \wedge ((c \bmod x) = 0) \wedge (b < c) \wedge (a < x)) \\ \rightarrow & (((a + b) < c) = \mathbf{t}) \end{aligned}$$

THEOREM: remainder-plus-fact

$$\begin{aligned} & (((b \bmod x) = 0) \wedge ((c \bmod x) = 0) \wedge (a < x)) \\ \rightarrow & (((a + b) \bmod c) = (a + (b \bmod c))) \end{aligned}$$

THEOREM: remainder-plus-times-times-proof

$$\begin{aligned} & (a < b) \\ \rightarrow & (((a + (b * c)) \bmod (b * d)) \\ & = (a + ((b * c) \bmod (b * d)))) \end{aligned}$$

THEOREM: remainder-plus-times-times

$$\begin{aligned} & (a < b) \\ \rightarrow & (((((a + (b * c)) \bmod (b * d)) \\ & = (a + ((b * c) \bmod (b * d)))) \\ \wedge & (((a + (c * b)) \bmod (d * b)) \\ & = (a + ((c * b) \bmod (d * b))))) \end{aligned}$$

THEOREM: remainder-plus-times-times-instance

$$\begin{aligned} & (a < b) \\ \rightarrow & (((((a + (b * c) + (b * d)) \bmod (b * e)) \\ & = (a + (b * ((c + d) \bmod e)))) \\ \wedge & (((a + (c * b) + (d * b)) \bmod (e * b)) \\ & = (a + (b * ((c + d) \bmod e))))) \end{aligned}$$

THEOREM: remainder-exp

$$(k \neq 0) \rightarrow ((\exp(n, k) \bmod n) = 0)$$

THEOREM: remainder-remainder

$$((b \bmod a) = 0) \rightarrow (((n \bmod b) \bmod a) = (n \bmod a))$$

THEOREM: remainder-1-arg1

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

THEOREM: remainder-1-arg2

$$(y \bmod 1) = 0$$

THEOREM: remainder-x-x

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

THEOREM: transitivity-of-divides

$$(((a \bmod b) = 0) \wedge ((b \bmod c) = 0)) \rightarrow ((a \bmod c) = 0)$$

DEFINITION:

double-number-induction (i, j)

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

THEOREM: remainder-exp-exp

if $j < i$ **then** **f**
 else **t** **endif**
 $\rightarrow ((\text{exp}(a, j) \bmod \text{exp}(a, i)) = 0)$

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

$(b = 1) \rightarrow ((a \div b) = \text{fix}(a))$

THEOREM: quotient-of-non-number

$(a \notin \mathbf{N}) \rightarrow ((a \div n) = (0 \div n))$

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

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

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

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

THEOREM: quotient-plus-proof
 $((b \bmod c) = 0) \rightarrow (((a + b) \div c) = ((a \div c) + (b \div c)))$

THEOREM: quotient-plus
 $((a \bmod c) = 0)$
 $\rightarrow (((((a + b) \div c) = ((a \div c) + (b \div c)))$
 $\quad \wedge (((b + a) \div c) = ((a \div c) + (b \div c)))$
 $\quad \wedge (((x + y + a) \div c)$
 $\quad \quad = (((x + y) \div c) + (a \div c))))$

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

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

THEOREM: quotient-times-proof
 $((a \bmod c) = 0) \rightarrow (((a * b) \div c) = (b * (a \div c)))$

THEOREM: quotient-times
 $((a \bmod c) = 0)$
 $\rightarrow (((((a * b) \div c) = (b * (a \div c)))$
 $\quad \wedge (((b * a) \div c) = (b * (a \div c))))$

THEOREM: quotient-times-instance

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

THEOREM: quotient-times-times-proof

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

THEOREM: quotient-times-times

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

THEOREM: quotient-difference1

$$\begin{aligned}
& ((a \bmod c) = (b \bmod c)) \\
\rightarrow \quad & (((a - b) \div c) = ((a \div c) - (b \div c)))
\end{aligned}$$

THEOREM: quotient-lessp-arg1

$$(a < b) \rightarrow ((a \div b) = 0)$$

THEOREM: quotient-difference2

$$\begin{aligned}
& (((a \bmod c) = 0) \wedge ((b \bmod c) \neq 0)) \\
\rightarrow \quad & (((a - b) \div c) \\
&= \text{if } b < a \text{ then } (a \div c) - (1 + (b \div c)) \\
&\quad \text{else } 0 \text{ endif})
\end{aligned}$$

THEOREM: quotient-difference3

$$\begin{aligned}
& (((b \bmod c) = 0) \wedge ((a \bmod c) \neq 0)) \\
\rightarrow \quad & (((a - b) \div c) \\
&= \text{if } b < a \text{ then } (a \div c) - (b \div c) \\
&\quad \text{else } 0 \text{ endif})
\end{aligned}$$

THEOREM: remainder-equals-its-first-argument

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

THEOREM: quotient-remainder-times

$$((x \bmod (a * b)) \div a) = ((x \div a) \bmod b)$$

THEOREM: quotient-remainder

$$((c \bmod a) = 0) \rightarrow (((b \bmod c) \div a) = ((b \div a) \bmod (c \div a)))$$

THEOREM: quotient-remainder-instance

$$((x \bmod (a * b)) \div a) = ((x \div a) \bmod b)$$

THEOREM: quotient-plus-fact

$$(((b \bmod x) = 0) \wedge ((c \bmod x) = 0) \wedge (a < x)) \\ \rightarrow (((a + b) \div c) = (b \div c))$$

THEOREM: quotient-plus-times-times-proof

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

THEOREM: quotient-plus-times-times

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

THEOREM: quotient-plus-times-times-instance

$$(a < b) \\ \rightarrow (((a + (b * c) + (b * d)) \div (b * e)) \\ = \text{if } b \simeq 0 \text{ then } 0 \\ \text{else } (c + d) \div e \text{ endif}) \\ \wedge (((a + (c * b) + (d * b)) \div (e * b)) \\ = \text{if } b \simeq 0 \text{ then } 0 \\ \text{else } (d + c) \div e \text{ endif}))$$

THEOREM: quotient-quotient

$$((b \div a) \div c) = (b \div (a * c))$$

THEOREM: quotient-exp

$$(k \neq 0) \\ \rightarrow ((\exp(n, k) \div n) \\ = \text{if } n \simeq 0 \text{ then } 0 \\ \text{else } \exp(n, k - 1) \text{ endif})$$

THEOREM: leq-quotient

$$(a < b) \\ \rightarrow \text{if } (b \div c) < (a \div c) \text{ then f} \\ \text{else t endif}$$

THEOREM: quotient-1-arg2

$$(n \div 1) = \text{fix}(n)$$

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

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

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

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

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

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

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

THEOREM: exp-difference

$$\begin{aligned} &(\text{if } b < c \text{ then } f \\ &\quad \text{else } t \text{ endif} \\ &\wedge (a \not\simeq 0)) \\ &\rightarrow (\exp(a, b - c) = (\exp(a, b) \div \exp(a, c))) \end{aligned}$$

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:

$$\begin{aligned} &\log(\text{base}, n) \\ &= \text{if } \text{base} < 2 \text{ then } 0 \\ &\quad \text{elseif } n \simeq 0 \text{ then } 0 \\ &\quad \text{else } 1 + \log(\text{base}, n \div \text{base}) \text{ endif} \end{aligned}$$

THEOREM: equal-log-0

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

THEOREM: log-0

$$(n \simeq 0) \rightarrow (\log(\text{base}, n) = 0)$$

THEOREM: log-1

$$(1 < \text{base}) \rightarrow (\log(\text{base}, 1) = 1)$$

DEFINITION:

$$\begin{aligned} &\text{double-log-induction}(\text{base}, a, b) \\ &= \text{if } \text{base} < 2 \text{ then } 0 \\ &\quad \text{elseif } a \simeq 0 \text{ then } 0 \\ &\quad \text{elseif } b \simeq 0 \text{ then } 0 \\ &\quad \text{else double-log-induction}(\text{base}, a \div \text{base}, b \div \text{base}) \text{ endif} \end{aligned}$$

THEOREM: leq-log-log

if $m < n$ **then** **f**
else **t** **endif**
 $\rightarrow$ **if** $\log(c, m) < \log(c, n)$ **then** **f**
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

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

THEOREM: log-quotient-exp

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

THEOREM: log-times-proof

$((1 < c) \wedge (n \neq 0)) \rightarrow (\log(c, c * n) = (1 + \log(c, n)))$

THEOREM: log-times

$((1 < c) \wedge (n \neq 0))$
 $\rightarrow ((\log(c, c * n) = (1 + \log(c, n)))$
 $\quad \wedge (\log(c, n * c) = (1 + \log(c, n))))$

THEOREM: log-times-exp-proof

$((1 < c) \wedge (n \neq 0)) \rightarrow (\log(c, n * \exp(c, m)) = (\log(c, n) + m))$

THEOREM: log-times-exp

$((1 < c) \wedge (n \neq 0))$
 $\rightarrow ((\log(c, n * \exp(c, m)) = (\log(c, n) + m))$
 $\quad \wedge (\log(c, \exp(c, m) * n) = (\log(c, n) + m)))$

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:

$\gcd(x, y)$

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

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

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

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

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

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

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

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

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

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

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

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

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

THEOREM: distributivity-of-times-over-gcd-proof

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

THEOREM: distributivity-of-times-over-gcd

$$\begin{aligned} &(\text{gcd}(x * z, y * z) = (z * \text{gcd}(x, y))) \\ \wedge &(\text{gcd}(z * x, y * z) = (z * \text{gcd}(x, y))) \\ \wedge &(\text{gcd}(x * z, z * y) = (z * \text{gcd}(x, y))) \\ \wedge &(\text{gcd}(z * x, z * y) = (z * \text{gcd}(x, y))) \end{aligned}$$

THEOREM: gcd-is-the-greatest

$$\begin{aligned} &((x \neq 0) \wedge (y \neq 0) \wedge ((x \bmod z) = 0) \wedge ((y \bmod z) = 0)) \\ \rightarrow &\text{if } \text{gcd}(x, y) < z \text{ then } \mathbf{f} \\ &\text{else } \mathbf{t} \text{ endif} \end{aligned}$$

THEOREM: common-divisor-divides-gcd

$$(((x \bmod z) = 0) \wedge ((y \bmod z) = 0)) \rightarrow ((\text{gcd}(x, y) \bmod z) = 0)$$

THEOREM: associativity-of-gcd-zero-case

$$((a \simeq 0) \vee (b \simeq 0) \vee (c \simeq 0)) \rightarrow (\text{gcd}(\text{gcd}(a, b), c) = \text{gcd}(a, \text{gcd}(b, c)))$$

THEOREM: associativity-of-gcd

$$\text{gcd}(\text{gcd}(a, b), c) = \text{gcd}(a, \text{gcd}(b, c))$$

THEOREM: commutativity2-of-gcd-zero-case

$$((a \simeq 0) \vee (b \simeq 0) \vee (c \simeq 0)) \rightarrow (\text{gcd}(b, \text{gcd}(a, c)) = \text{gcd}(a, \text{gcd}(b, c)))$$

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

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

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:

$$\text{integerp}(x) = ((x \in \mathbf{N}) \vee (\text{negativep}(x) \wedge (\text{negative-guts}(x) \neq 0)))$$

DEFINITION:

$\text{fix-int}(x)$
= **if** $\text{integerp}(x)$ **then** x
 else 0 **endif**

DEFINITION:

$\text{izerop}(i)$
= **if** $\text{integerp}(i)$ **then** $i = 0$
 else $\mathbf{t}$ **endif**

DEFINITION:

$\text{ilessp}(i, j)$
= **if** $\text{negativep}(i)$
 then if $\text{negativep}(j)$ **then** $\text{negative-guts}(j) < \text{negative-guts}(i)$
 else $\mathbf{t}$ **endif**
 elseif $\text{negativep}(j)$ **then** $\mathbf{f}$
 else $i < j$ **endif**

DEFINITION: $\text{ileq}(i, j) = (\text{ilessp}(i, j) \vee (i = j))$

DEFINITION:

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

DEFINITION:

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

DEFINITION: $\text{idifference}(x, y) = \text{iplus}(x, \text{ineg}(y))$

DEFINITION:

$\text{iabs}(i)$
= **if** $\text{negativep}(i)$ **then** $\text{negative-guts}(i)$
 else $\text{fix}(i)$ **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)) = neg (x)

THEOREM: fix-int-iabs

fix-int (iabs (x)) = iabs (x)

THEOREM: fix-int-itimes

fix-int (itimes (x, y)) = itimes (x, y)

THEOREM: neg-id

($\neg$ integerp (x)) $\rightarrow$ (neg (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

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

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

$$\text{iplus}(a, \text{ineg}(a)) = 0$$

THEOREM: iplus-ineg2

$$\text{iplus}(\text{ineg}(a), a) = 0$$

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
 $\text{idifference}(x, x) = 0$

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

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

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

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

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

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

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

DEFINITION:

```

cancel-iplus( $x$ )
=  if listp( $x$ )  $\wedge$  (car( $x$ ) = 'equal)
    then if listp(cadr( $x$ ))
         $\wedge$  (caadr( $x$ ) = 'iplus)
         $\wedge$  listp(caddr( $x$ ))
         $\wedge$  (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( $x$ ))
             $\wedge$  (caadr( $x$ ) = 'iplus)
             $\wedge$  (caddr( $x$ )  $\in$  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( $x$ ))
             $\wedge$  (caaddr( $x$ ) = 'iplus)
             $\wedge$  (cadr( $x$ )  $\in$  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

$(\text{listp}(x) \wedge (\text{car}(x) = \text{'iplus})) \rightarrow \text{integerp}(\text{eval}(\mathbf{t}, x, a))$

THEOREM: integerp-eval\$-iplus-tree

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

THEOREM: integerp-eval\$-bridge

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

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 ∈ 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)
→  (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

```
(¬ integerp (a)) → ((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) ∧ (car (x) = 'idifference)
    then if listp (cadr (x))
        ∧ (caadr (x) = 'iplus)
        ∧ listp (caddr (x))
        ∧ (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)),
```

```

                                                    iplus-fringe (caddr (x))))))
elseif listp (cadr (x))
     $\wedge$  (caadr (x) = 'iplus)
     $\wedge$  (caddr (x)  $\in$  iplus-fringe (cadr (x)))
then iplus-tree (delete (caddr (x), iplus-fringe (cadr (x))))
elseif listp (caddr (x))
     $\wedge$  (caaddr (x) = 'iplus)
     $\wedge$  (cadr (x)  $\in$  iplus-fringe (caddr (x)))
then list ('ineg,
            iplus-tree (delete (cadr (x), iplus-fringe (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
 $\text{izerop}(x) \rightarrow (\text{itimes}(x, y) = 0)$

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

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
 $(\text{itimes}(x, \text{iplus}(y, z)) = \text{iplus}(\text{itimes}(x, y), \text{itimes}(x, z)))$
 $\wedge (\text{itimes}(\text{iplus}(x, y), z) = \text{iplus}(\text{itimes}(x, z), \text{itimes}(y, z)))$

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

$$\begin{aligned} & (\text{itimes}(\text{idifference}(a, b), c) = \text{idifference}(\text{itimes}(a, c), \text{itimes}(b, c))) \\ & \wedge (\text{itimes}(a, \text{idifference}(b, c)) = \text{idifference}(\text{itimes}(a, b), \text{itimes}(a, c))) \end{aligned}$$

THEOREM: equal-itimes-0

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

THEOREM: equal-itimes-1

$$\begin{aligned} & (\text{itimes}(a, b) = 1) \\ & = (((a = 1) \wedge (b = 1)) \vee ((a = -1) \wedge (b = -1))) \end{aligned}$$

THEOREM: equal-itimes-minus-1

$$\begin{aligned} & (\text{itimes}(a, b) = -1) \\ & = (((a = -1) \wedge (b = 1)) \vee ((a = 1) \wedge (b = -1))) \end{aligned}$$

THEOREM: itimes-1-arg1

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

THEOREM: quotient-remainder-uniqueness

$$\begin{aligned} & ((a = (r + (b * q))) \wedge (r < b)) \\ & \rightarrow ((\text{fix}(r) = (a \bmod b)) \wedge (\text{fix}(q) = (a \div b))) \end{aligned}$$

DEFINITION:

iquotient(i, j)

$$\begin{aligned} & = \text{if } \text{izerop}(j) \text{ then } 0 \\ & \quad \text{elseif } \text{negativep}(i) \\ & \quad \text{then if } \text{negativep}(j) \\ & \quad \quad \text{then if } (\text{negative-guts}(i) \bmod \text{negative-guts}(j)) = 0 \\ & \quad \quad \quad \text{then } \text{negative-guts}(i) \div \text{negative-guts}(j) \\ & \quad \quad \quad \text{else } 1 + (\text{negative-guts}(i) \div \text{negative-guts}(j)) \text{ endif} \\ & \quad \quad \text{elseif } (\text{negative-guts}(i) \bmod j) = 0 \\ & \quad \quad \quad \text{then } \text{fix-int}(-(\text{negative-guts}(i) \div j)) \\ & \quad \quad \quad \text{else } \text{fix-int}(-(1 + (\text{negative-guts}(i) \div j))) \text{ endif} \\ & \quad \text{elseif } \text{negativep}(j) \text{ then } \text{fix-int}(-(i \div \text{negative-guts}(j))) \\ & \quad \text{else } i \div j \text{ endif} \end{aligned}$$

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

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

THEOREM: division-theorem-part3

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

THEOREM: division-theorem

$$\begin{aligned} & (\text{integerp}(i) \wedge \text{integerp}(j) \wedge (j \neq 0)) \\ \rightarrow & ((\text{iplus}(\text{iremainder}(i, j), \text{itimes}(j, \text{iquotient}(i, j))) = i) \\ & \wedge \text{ileq}(0, \text{iremainder}(i, j)) \\ & \wedge \text{ilessp}(\text{iremainder}(i, j), \text{iabs}(j))) \end{aligned}$$

THEOREM: quotient-difference-lessp-arg2

$$\begin{aligned} & (((a \bmod c) = 0) \wedge (b < c)) \\ \rightarrow & (((a - b) \div c) \\ & = \text{if } b \simeq 0 \text{ then } a \div c \\ & \quad \text{elseif } b < a \text{ then } (a \div c) - (1 + (b \div c)) \\ & \quad \text{else } 0 \text{ endif}) \end{aligned}$$

THEOREM: iquotient-iremainder-uniqueness

$$\begin{aligned} & (\text{integerp}(i) \\ & \wedge \text{integerp}(j) \\ & \wedge \text{integerp}(r) \\ & \wedge \text{integerp}(q) \\ & \wedge (j \neq 0) \\ & \wedge (i = \text{iplus}(r, \text{itimes}(j, q))) \\ & \wedge \text{ileq}(0, r) \\ & \wedge \text{ilessp}(r, \text{iabs}(j))) \\ \rightarrow & ((r = \text{iremainder}(i, j)) \wedge (q = \text{iquotient}(i, j))) \end{aligned}$$

DEFINITION:

$$\begin{aligned} & \text{idiv}(i, j) \\ = & \text{if } \text{izerop}(j) \text{ then } 0 \\ & \text{elseif } \text{negativep}(i) \\ & \quad \text{then if } \text{negativep}(j) \text{ then } \text{negative-guts}(i) \div \text{negative-guts}(j) \\ & \quad \quad \text{elseif } (\text{negative-guts}(i) \bmod j) = 0 \\ & \quad \quad \text{then } \text{fix-int}(-(\text{negative-guts}(i) \div j)) \\ & \quad \quad \text{else } \text{fix-int}(-(1 + (\text{negative-guts}(i) \div j))) \text{ endif} \\ & \quad \text{elseif } \text{negativep}(j) \\ & \quad \text{then if } (i \bmod \text{negative-guts}(j)) = 0 \\ & \quad \quad \text{then } \text{fix-int}(-(i \div \text{negative-guts}(j))) \\ & \quad \quad \text{else } \text{fix-int}(-(1 + (i \div \text{negative-guts}(j)))) \text{ endif} \\ & \quad \text{else } i \div j \text{ endif} \end{aligned}$$

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

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

THEOREM: division-theorem-for-truncate-to-neginf-part3

(integerp (j) $\wedge$ ilessp (j, 0))
 $\rightarrow$ (ileq (imod (i, j), 0) $\wedge$ ilessp (j, imod (i, j)))

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

(integerp (i) $\wedge$ integerp (j) $\wedge$ (j $\neq$ 0))
 $\rightarrow$ ((iplus (imod (i, j), itimes (j, idiv (i, j))) = i)
 $\wedge$ **if** ilessp (0, j)
then ileq (0, imod (i, j)) $\wedge$ ilessp (imod (i, j), j)
else ileq (imod (i, j), 0) $\wedge$ ilessp (j, imod (i, j)) **endif**)

THEOREM: idiv-imod-uniqueness

(integerp (i)
 $\wedge$ integerp (j)
 $\wedge$ integerp (r)
 $\wedge$ integerp (q)
 $\wedge$ (j $\neq$ 0)
 $\wedge$ (i = iplus (r, itimes (j, q)))
 $\wedge$ **if** ilessp (0, j) **then** ileq (0, r) $\wedge$ ilessp (r, j)
else ileq (r, 0) $\wedge$ ilessp (j, r) **endif**)
 $\rightarrow$ ((r = imod (i, j)) $\wedge$ (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) $\rightarrow$ (iplus (irem (i, j), itimes (j, iquo (i, j))) = i)

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

(integerp (i) $\wedge$ integerp (j) $\wedge$ (j $\neq$ 0) $\wedge$ ileq (0, i))
 $\rightarrow$ (ileq (0, irem (i, j)) $\wedge$ ilessp (irem (i, j), iabs (j)))

THEOREM: division-theorem-for-truncate-to-zero-part3

(integerp (i) $\wedge$ integerp (j) $\wedge$ (j $\neq$ 0) $\wedge$ ilessp (i, 0))
 $\rightarrow$ (ileq (irem (i, j), 0) $\wedge$ ilessp (ineg (iabs (j)), irem (i, j)))

THEOREM: division-theorem-for-truncate-to-zero

$$\begin{aligned}
& (\text{integerp}(i) \wedge \text{integerp}(j) \wedge (j \neq 0)) \\
\rightarrow & ((\text{ipplus}(\text{irem}(i, j), \text{itimes}(j, \text{iquo}(i, j))) = i) \\
& \wedge \text{if ileq}(0, i) \\
& \quad \text{then ileq}(0, \text{irem}(i, j)) \wedge \text{ilessp}(\text{irem}(i, j), \text{iabs}(j)) \\
& \quad \text{else ileq}(\text{irem}(i, j), 0) \\
& \quad \wedge \text{ilessp}(\text{ineg}(\text{iabs}(j)), \text{irem}(i, j)) \text{endif})
\end{aligned}$$

THEOREM: iquo-irem-uniqueness

$$\begin{aligned}
& (\text{integerp}(i) \\
& \wedge \text{integerp}(j) \\
& \wedge \text{integerp}(r) \\
& \wedge \text{integerp}(q) \\
& \wedge (j \neq 0) \\
& \wedge (i = \text{ipplus}(r, \text{itimes}(j, q))) \\
& \wedge \text{if ileq}(0, i) \text{ then ileq}(0, r) \wedge \text{ilessp}(r, \text{iabs}(j)) \\
& \quad \text{else ileq}(r, 0) \wedge \text{ilessp}(\text{ineg}(\text{iabs}(j)), r) \text{endif}) \\
\rightarrow & ((r = \text{irem}(i, j)) \wedge (q = \text{iquo}(i, j)))
\end{aligned}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 illessp; name this event ‘illessp-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 caddr (x)  $\simeq$  nil then list ('times, car (x), cadr (x))
   else list ('times, car (x), times-tree (cdr (x))) endif
```

DEFINITION:

```
times-fringe (x)
=  if listp (x)  $\wedge$  (car (x) = 'times)
   then append (times-fringe (cadr (x)), times-fringe (caddr (x)))
   else list (x) endif
```

DEFINITION:

```
or-zero-tree (x)
=  if x  $\simeq$  nil then ' (false)
   elseif cdr (x)  $\simeq$  nil then list ('zerop, car (x))
   elseif caddr (x)  $\simeq$  nil
   then list ('or, list ('zerop, car (x)), list ('zerop, cadr (x)))
   else list ('or, list ('zerop, car (x)), or-zero-tree (cdr (x))) endif
```

DEFINITION:

```
and-not-zero-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-zero-tree (cdr (x))) endif
```

THEOREM: numberp-eval\$-times

$(\text{car}(x) = \text{'times}) \rightarrow (\text{eval}\$(\mathbf{t}, x, a) \in \mathbf{N})$

THEOREM: eval\$-times

$\text{eval}\$(\mathbf{t}, \text{list}(\text{'times}, x, y), a) = (\text{eval}\$(\mathbf{t}, x, a) * \text{eval}\$(\mathbf{t}, y, a))$

THEOREM: eval\$-times2

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

THEOREM: eval\$-or

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

THEOREM: eval\$-or2

$$(\text{car}(x) = 'or)$$

$$\rightarrow (\text{eval\$}(\mathbf{t}, x, a) = (\text{eval\$}(\mathbf{t}, \text{cadr}(x), a) \vee \text{eval\$}(\mathbf{t}, \text{caddr}(x), a)))$$

THEOREM: eval\$-equal

$$(\text{car}(x) = 'equal)$$

$$\rightarrow (\text{eval\$}(\mathbf{t}, x, a) = (\text{eval\$}(\mathbf{t}, \text{cadr}(x), a) = \text{eval\$}(\mathbf{t}, \text{caddr}(x), a)))$$

THEOREM: eval\$-lessp

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

THEOREM: eval\$-lessp2

$$(\text{car}(x) = 'lessp)$$

$$\rightarrow (\text{eval\$}(\mathbf{t}, x, a) = (\text{eval\$}(\mathbf{t}, \text{cadr}(x), a) < \text{eval\$}(\mathbf{t}, \text{caddr}(x), a)))$$

THEOREM: eval\$-quotient

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

THEOREM: eval\$-quotient2

$$(\text{car}(x) = 'quotient)$$

$$\rightarrow (\text{eval\$}(\mathbf{t}, x, a) = (\text{eval\$}(\mathbf{t}, \text{cadr}(x), a) \div \text{eval\$}(\mathbf{t}, \text{caddr}(x), a)))$$

THEOREM: eval\$-if

$$\text{eval\$}(\mathbf{t}, \text{list}('if, x, y, z), a)$$

$$= \text{if } \text{eval\$}(\mathbf{t}, x, a) \text{ then } \text{eval\$}(\mathbf{t}, y, a) \\ \text{else } \text{eval\$}(\mathbf{t}, z, a) \text{ endif}$$

THEOREM: eval\$-if2

$$(\text{car}(x) = 'if)$$

$$\rightarrow (\text{eval\$}(\mathbf{t}, x, a) \\ = \text{if } \text{eval\$}(\mathbf{t}, \text{cadr}(x), a) \text{ then } \text{eval\$}(\mathbf{t}, \text{caddr}(x), a) \\ \text{else } \text{eval\$}(\mathbf{t}, \text{caddr}(x), a) \text{ endif})$$

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

$$(a \not\leq 0) \rightarrow (((a * x) \not\leq (a * y)) = (x \not\leq y))$$

THEOREM: stupid-hack

$$((a \in \mathbf{N}) \wedge (b \in \mathbf{N})) \rightarrow (((a \not\leq b) \wedge (b \not\leq a)) = (a = b))$$

THEOREM: equal-times-arg1

$$(a \neq 0) \rightarrow (((a * x) = (a * y)) = (\text{fix}(x) = \text{fix}(y)))$$

THEOREM: equal-times-bridge

$$((a * b) = (c * a * d)) = ((a \simeq 0) \vee (\text{fix}(b) = (c * d)))$$

THEOREM: eval\$-times-member

$$\begin{aligned} & (e \in x) \\ \rightarrow & (\text{eval}\$(\mathbf{t}, \text{times-tree}(x), a) \\ & = (\text{eval}\$(\mathbf{t}, e, a) * \text{eval}\$(\mathbf{t}, \text{times-tree}(\text{delete}(e, x)), a))) \end{aligned}$$

THEOREM: zerop-makes-times-tree-zero

$$\begin{aligned} & ((\neg \text{eval}\$(\mathbf{t}, \text{and-not-zerop-tree}(x), a)) \wedge \text{subbagp}(x, y)) \\ \rightarrow & (\text{eval}\$(\mathbf{t}, \text{times-tree}(y), a) = 0) \end{aligned}$$

THEOREM: or-zerop-tree-is-not-zerop-tree

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

THEOREM: zerop-makes-times-tree-zero2

$$\begin{aligned} & (\text{eval}\$(\mathbf{t}, \text{or-zerop-tree}(x), a) \wedge \text{subbagp}(x, y)) \\ \rightarrow & (\text{eval}\$(\mathbf{t}, \text{times-tree}(y), a) = 0) \end{aligned}$$

THEOREM: times-tree-append

$$\begin{aligned} & \text{eval}\$(\mathbf{t}, \text{times-tree}(\text{append}(x, y)), a) \\ = & (\text{eval}\$(\mathbf{t}, \text{times-tree}(x), a) * \text{eval}\$(\mathbf{t}, \text{times-tree}(y), a)) \end{aligned}$$

THEOREM: times-tree-of-times-fringe

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

DEFINITION:

cancel-lessp-times(x)

```
=  if (car(x) = 'lessp)
    and (cadr(x) = 'times)
    and (caddr(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

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

THEOREM: zerop-makes-lessp-false-bridge

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

THEOREM: correctness-of-cancel-lessp-times

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

DEFINITION:
cancel-quotient-times(x)

$$\begin{aligned} = & \text{if}(\text{car}(x) = \text{'quotient}) \\ & \wedge (\text{caadr}(x) = \text{'times}) \\ & \wedge (\text{caaddr}(x) = \text{'times}) \\ & \text{then if listp}(\text{bagint}(\text{times-fringe}(\text{cadr}(x)), \text{times-fringe}(\text{caddr}(x)))) \\ & \quad \text{then cons}(\text{'if}, \\ & \quad \quad \text{cons}(\text{and-not-zerop-tree}(\text{bagint}(\text{times-fringe}(\text{cadr}(x)), \\ & \quad \quad \quad \text{times-fringe}(\text{caddr}(x)))), \\ & \quad \quad \text{cons}(\text{list}(\text{'quotient}, \\ & \quad \quad \quad \text{times-tree}(\text{bagdiff}(\text{times-fringe}(\text{cadr}(x)), \\ & \quad \quad \quad \quad \text{bagint}(\text{times-fringe}(\text{cadr}(x)), \\ & \quad \quad \quad \quad \quad \text{times-fringe}(\text{caddr}(x)))))), \\ & \quad \quad \text{times-tree}(\text{bagdiff}(\text{times-fringe}(\text{caddr}(x)), \\ & \quad \quad \quad \text{bagint}(\text{times-fringe}(\text{cadr}(x)), \\ & \quad \quad \quad \quad \text{times-fringe}(\text{caddr}(x)))))), \\ & \quad \quad \text{'((zero))})) \\ & \text{else } x \text{ endif} \\ & \text{else } x \text{ endif} \end{aligned}$$

THEOREM: zerop-makes-quotient-zero-bridge

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

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

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

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

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)

```

=  if (car( $x$ ) = 'equal)
    ^  (cadr( $x$ ) = 'times)
    ^  (caaddr( $x$ ) = 'times)
    then if listp(bagint(times-fringe(cadr( $x$ )), times-fringe(caddr( $x$ ))))
        then list('or,
                    or-zerop-tree(bagint(times-fringe(cadr( $x$ )),
                                         times-fringe(caddr( $x$ ))),
                    list('equal,
                        times-tree(bagdiff(times-fringe(cadr( $x$ )),
                                           bagint(times-fringe(cadr( $x$ )),
                                           times-fringe(caddr( $x$ ))))),
                        times-tree(bagdiff(times-fringe(caddr( $x$ )),
                                           bagint(times-fringe(cadr( $x$ )),
                                           times-fringe(caddr( $x$ ))))))
                    else  $x$  endif
    else  $x$  endif

```

THEOREM: zerop-makes-equal-true-bridge

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

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

$$\begin{aligned}
&(\text{subbagp}(x, y) \wedge \text{subbagp}(x, z) \wedge (\neg \text{eval}\$ (\mathbf{t}, \text{or-zerop-tree}(x), a))) \\
&\rightarrow ((\text{eval}\$ (\mathbf{t}, \text{times-tree}(\text{bagdiff}(y, x)), a) \\
&\quad = \text{eval}\$ (\mathbf{t}, \text{times-tree}(\text{bagdiff}(z, x)), a)) \\
&\quad = (\text{eval}\$ (\mathbf{t}, \text{times-tree}(y), a) = \text{eval}\$ (\mathbf{t}, \text{times-tree}(z), a)))
\end{aligned}$$

THEOREM: cancel-equal-times-preserves-inequality

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

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

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

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:
 $\text{rationalp}(x)$
 $= (\text{rational-formp}(x))$

$\wedge$ integerp (numerator (x))
 $\wedge$ (denominator (x) $\neq 0$)

DEFINITION:

fix-rational (x)
 = **if** rationalp (x) **then** x
 else rational (0, 1) **endif**

DEFINITION:

reduce (r)
 = **if** rationalp (r)
 then if negativep (numerator (r))
 then rational ($-$ (negative-guts (numerator (r))
 $\div$ gcd (negative-guts (numerator (r)),
 denominator (r))),
 denominator (r)
 $\div$ gcd (negative-guts (numerator (r)),
 denominator (r)))
 else rational (numerator (r)
 $\div$ gcd (numerator (r), denominator (r)),
 denominator (r)
 $\div$ gcd (numerator (r), denominator (r))) **endif**
 else rational (0, 1) **endif**

DEFINITION:

simple-rplus (x, y)
 = rational (iplus (itimes (numerator (fix-rational (x)),
 denominator (fix-rational (y))),
 itimes (numerator (fix-rational (y)),
 denominator (fix-rational (x))),
 itimes (denominator (fix-rational (x)),
 denominator (fix-rational (y))))

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

DEFINITION:

simple-rneg (x)
 = rational (ineg (numerator (fix-rational (x))),
 denominator (fix-rational (x)))

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

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

DEFINITION:

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

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

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

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

DEFINITION: $\text{rinverse}(r) = \text{reduce}(\text{simple-rinverse}(r))$

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

DEFINITION:
 $\text{simple-rmagnitude}(x)$
 $= \text{if } \text{negativep}(\text{numerator}(\text{fix-rational}(x))) \text{ then } \text{rneg}(\text{fix-rational}(x))$
 $\quad \text{else } \text{fix-rational}(x) \text{ endif}$

DEFINITION: $\text{rmagnitude}(x) = \text{reduce}(\text{simple-rmagnitude}(x))$

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

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

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

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

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

THEOREM: itimes-ineg-arg2

$$\text{itimes}(x, \text{ineg}(y)) = \text{ineg}(\text{itimes}(x, y))$$

THEOREM: integerp-if-negativep-non-zero

$$(\text{negativep}(x) \wedge (x \neq (-\ 0))) \rightarrow \text{integerp}(x)$$

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

$$(\text{integerp}(x) \wedge \text{integerp}(y)) \rightarrow ((\text{ineg}(x) = \text{ineg}(y)) = (x = y))$$

THEOREM: itimes-is-times

$$((x \in \mathbf{N}) \wedge (y \in \mathbf{N})) \rightarrow (\text{itimes}(x, y) = (x * y))$$

THEOREM: iplus-is-plus

$$((x \in \mathbf{N}) \wedge (y \in \mathbf{N})) \rightarrow (\text{iplus}(x, y) = (x + y))$$

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
 $(\text{rationalp}(x) \rightarrow \text{integerp}(\text{numerator}(x)))$
 $\wedge (\text{rationalp}(x) \rightarrow (\text{denominator}(x) \in \mathbf{N}))$
 $\wedge (\text{rationalp}(x) \rightarrow (\text{denominator}(x) \neq 0))$

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

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

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

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

$$((0 < a) \wedge (0 < b)) \rightarrow (\gcd(a \div \gcd(a, b), b \div \gcd(a, b)) = 1)$$

THEOREM: divides-each-equality

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

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

EVENT: Disable illessp; name this event 'illessp-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 ≅ 0 then '(0 . 1)
    elseif y ≅ 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

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

$$\begin{aligned}
& \text{itimes}(\text{cdr}(\text{gcd-factors}(x, y)), y)) \\
= & \text{gcd}(x, y)
\end{aligned}$$

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

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

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

$$\begin{aligned}
& ((x \in \mathbf{N}) \wedge (y \in \mathbf{N})) \\
\rightarrow & (\text{gcd}(x, y) \\
& = \text{iplus}(\text{itimes}(\text{car}(\text{gcd-factors}(x, y)), x), \\
& \quad \text{itimes}(\text{cdr}(\text{gcd-factors}(x, y)), y)))
\end{aligned}$$

THEOREM: dpr-hack1

$$\begin{aligned}
& \text{itimes}(x, \text{iplus}(\text{itimes}(c, a), \text{itimes}(b, q))) \\
= & \text{iplus}(\text{itimes}(c, \text{itimes}(a, x)), \text{itimes}(b, \text{itimes}(q, x)))
\end{aligned}$$

THEOREM: dpr-hack2

$$\begin{aligned}
& ((x \in \mathbf{N}) \wedge (y \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (\text{iremainder}(\text{itimes}(c, y), x) = 0)) \\
\rightarrow & (\text{itimes}(\text{iquotient}(\text{itimes}(c, y), x), x) = \text{itimes}(c, y))
\end{aligned}$$

THEOREM: dpr-hack3

$$\begin{aligned}
& \text{iplus}(\text{itimes}(c, \text{itimes}(a, x)), \text{itimes}(b, \text{itimes}(c, y))) \\
= & \text{iplus}(\text{itimes}(c, \text{itimes}(a, x)), \text{itimes}(c, \text{itimes}(b, y)))
\end{aligned}$$

THEOREM: dpr-hack4

$$\begin{aligned}
& \text{iplus}(\text{itimes}(c, \text{itimes}(a, x)), \text{itimes}(c, \text{itimes}(b, y))) \\
= & \text{itimes}(c, \text{iplus}(\text{itimes}(a, x), \text{itimes}(b, y)))
\end{aligned}$$

THEOREM: dpr-hack5

$$\begin{aligned}
& ((x \in \mathbf{N}) \wedge (y \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (\text{iremainder}(\text{itimes}(c, y), x) = 0)) \\
\rightarrow & (\text{itimes}(x, \text{iplus}(\text{itimes}(c, a), \text{itimes}(b, \text{iquotient}(\text{itimes}(c, y), x)))) \\
& = \text{itimes}(c, \text{iplus}(\text{itimes}(a, x), \text{itimes}(b, y))))
\end{aligned}$$

THEOREM: remainder-0-sufficiency

$$\begin{aligned}
& ((x \in \mathbf{N}) \wedge (c \in \mathbf{N}) \wedge (\text{itimes}(x, p) = c)) \\
\rightarrow & (((c \bmod x) = 0) = \mathbf{t})
\end{aligned}$$

THEOREM: divides-product-reduction

$$\begin{aligned}
& ((x \in \mathbf{N}) \\
& \wedge (y \in \mathbf{N}) \\
& \wedge (c \in \mathbf{N}) \\
& \wedge (\text{gcd}(x, y) = 1) \\
& \wedge (((c * y) \bmod x) = 0)) \\
\rightarrow & ((c \bmod x) = 0)
\end{aligned}$$

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 illessp; name this event ‘illessp-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
 $(\text{gcd}(a, b) = 1) \rightarrow (((b * c) \bmod a) = 0) = ((c \bmod a) = 0)$

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

THEOREM: equal-times-times-quotient-arg2
 $((b \bmod e) = 0) \wedge ((d \bmod e) = 0)$
 $\rightarrow (((a * (b \div e)) = (c * (d \div e)))$
 $= ((a * b) = (c * d)))$
 $\wedge (((b \div e) * a) = (c * (d \div e)))$
 $= ((a * b) = (c * d)))$
 $\wedge (((b \div e) * a) = ((d \div e) * c))$
 $= ((a * b) = (c * d)))$

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

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

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

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

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

THEOREM: quotient-gcd-times-fact5

$$\begin{aligned} & (((v * z) = (w * x)) \wedge (0 < v) \wedge (0 < x)) \\ \rightarrow & (((v \div \gcd(v, w)) = (x \div \gcd(x, z))) = \mathbf{t}) \end{aligned}$$

THEOREM: equal-times-gcd-bridge1

$$\begin{aligned} & (((b \div \gcd(c, b)) = (d \div \gcd(a, d))) \\ \wedge & ((c \div \gcd(c, b)) = (a \div \gcd(a, d)))) \\ \rightarrow & (((a * b) = (c * d)) = \mathbf{t}) \end{aligned}$$

THEOREM: numberp-if-integerp-and-not-negativep

$$(\text{integerp}(x) \wedge (\neg \text{negativep}(x))) \rightarrow (x \in \mathbf{N})$$

THEOREM: negativep-if-integerp-and-not-numberp

$$(\text{integerp}(x) \wedge (x \notin \mathbf{N})) \rightarrow \text{negativep}(x)$$

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

$$\begin{aligned} & \text{rplus}(a, b) \\ = & \text{if rationalp}(a) \\ & \text{then if rationalp}(b) \\ & \text{then reduce}(\text{rational}(\text{iplus}(\text{itimes}(\text{numerator}(a), \text{denominator}(b))), \end{aligned}$$

```

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

THEOREM: rplus-requal-arg1
 $\text{requal}(a, b) \rightarrow \text{requal}(\text{rplus}(a, x), \text{rplus}(b, x))$

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

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

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

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

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

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

THEOREM: reduce-nrationalp
 $(\neg \text{rationalp}(x)) \rightarrow (\text{reduce}(x) = \text{rational}(0, 1))$

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

THEOREM: negativep-ineg
 $\text{negativep}(\text{ineg}(x)) = (0 < x)$

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

THEOREM: rational-ineg-numerator-reduce-bridge
 $\text{rational}(\text{ineg}(\text{numerator}(\text{reduce}(x))), \text{denominator}(\text{reduce}(x)))$
 $= \text{reduce}(\text{rational}(\text{ineg}(\text{numerator}(x)), \text{denominator}(x)))$

THEOREM: reduce-0

$$\text{reduce}(\text{rational}(0, x)) = \text{rational}(0, 1)$$

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

$$\begin{aligned} &\text{rationalp}(x) \\ \rightarrow &(\text{rational-formp}(x) \\ &\wedge \text{integerp}(\text{numerator}(x)) \\ &\wedge (0 < \text{denominator}(x))) \end{aligned}$$

THEOREM: means-rationalp

$$(\text{integerp}(n) \wedge (0 < d)) \rightarrow \text{rationalp}(\text{rational}(n, d))$$

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
 $((a * b) = (c * d)) \wedge ((a * x) = (c * y)) \wedge (a \neq 0)$
 $\rightarrow ((b * y) = (d * x)) = \mathbf{t}$

THEOREM: equal-times-bridge2
 $((a * b) = (c * d)) \wedge ((a * x) = (c * y)) \wedge (a \neq 0)$
 $\rightarrow ((y * b) = (d * x)) = \mathbf{t}$

THEOREM: equal-times-bridge3
 $((a * b) = (c * d)) \wedge ((a * x) = (c * y)) \wedge (a \neq 0)$
 $\rightarrow ((b * y) = (x * d)) = \mathbf{t}$

THEOREM: equal-times-bridge4
 $((a * b) = (c * d)) \wedge ((a * x) = (c * y)) \wedge (a \neq 0)$
 $\rightarrow ((y * b) = (x * d)) = \mathbf{t}$

THEOREM: transitivity-of-requal-bridge
 $(\text{requal}(a, b) \wedge \text{requal}(b, c)) \rightarrow \text{requal}(a, c)$

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

THEOREM: transitivity-of-requal
 $(\text{requal}(a, b) \wedge \text{requal}(b, c)) \rightarrow \text{requal}(a, c)$
 $\wedge ((\text{requal}(a, b) \wedge \text{requal}(c, b)) \rightarrow \text{requal}(a, c))$
 $\wedge ((\text{requal}(b, a) \wedge \text{requal}(b, c)) \rightarrow \text{requal}(a, c))$
 $\wedge ((\text{requal}(b, a) \wedge \text{requal}(c, b)) \rightarrow \text{requal}(a, c))$

THEOREM: equal-requal-rewrite
 $(\text{requal}(b, c) \rightarrow (\text{requal}(a, b) = \text{requal}(a, c)))$
 $\wedge (\text{requal}(b, c) \rightarrow (\text{requal}(a, b) = \text{requal}(c, a)))$
 $\wedge (\text{requal}(b, c) \rightarrow (\text{requal}(b, a) = \text{requal}(c, a)))$

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

THEOREM: requal-simple-rplus-bridge

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

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
 $\text{requal}(\text{simple-rplus}(\text{simple-rneg}(x), \text{simple-rneg}(y)), \text{simple-rneg}(\text{simple-rplus}(x, y)))$

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

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

THEOREM: rdifference-rdifference-arg1
 $\text{rdifference}(\text{rdifference}(x, y), z) = \text{rdifference}(x, \text{rplus}(y, z))$

THEOREM: rdifference-rdifference-arg2
 $\text{rdifference}(x, \text{rdifference}(y, z)) = \text{rdifference}(\text{rplus}(x, z), y)$

THEOREM: rplus-rneg-arg1
 $\text{rplus}(\text{rneg}(x), y) = \text{rdifference}(y, x)$

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

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

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

THEOREM: simple-rtimes-rzerop
 $\text{rzerop}(x) \rightarrow ((\text{numerator}(\text{simple-rtimes}(x, y)) = 0) \wedge (\text{numerator}(\text{simple-rtimes}(y, x)) = 0))$

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

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

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

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

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
 $(\text{requal}(\text{simple-rtimes}(\text{reduce}(x), y), z) = \text{requal}(\text{simple-rtimes}(x, y), z))$
 $\wedge (\text{requal}(\text{simple-rtimes}(x, \text{reduce}(y)), z)$
 $\quad = \text{requal}(\text{simple-rtimes}(x, y), z))$
 $\wedge (\text{requal}(z, \text{simple-rtimes}(\text{reduce}(x), y))$
 $\quad = \text{requal}(z, \text{simple-rtimes}(x, y)))$
 $\wedge (\text{requal}(z, \text{simple-rtimes}(x, \text{reduce}(y)))$
 $\quad = \text{requal}(z, \text{simple-rtimes}(x, y)))$

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

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

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
 $((\neg \text{rzerop}(x)) \wedge (\neg \text{rzerop}(y)) \wedge (\neg \text{rzerop}(z)))$
 $\rightarrow \text{requal}(\text{simple-rtimes}(\text{simple-rplus}(x, y), z),$
 $\quad \text{simple-rplus}(\text{simple-rtimes}(x, z), \text{simple-rtimes}(y, z)))$

THEOREM: requal-rzerop-simple-rplus
 $\text{rzerop}(x)$
 $\rightarrow (\text{requal}(\text{simple-rplus}(x, y), \text{fix-rational}(y))$
 $\quad \wedge \text{requal}(\text{simple-rplus}(y, x), \text{fix-rational}(y)))$

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

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

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

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

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

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

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

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

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

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

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

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

THEOREM: rn timer-neg-rdifference
 $\text{rn timer}(\text{rdifference}(x, y)) = \text{rdifference}(y, x)$

THEOREM: rdifference-rneg-arg2
 $\text{rdifference}(x, \text{rn timer}(y)) = \text{rplus}(x, y)$

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 caddr( $x$ )  $\simeq$  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

$$((v \not\approx 0) \wedge (z \in \mathbf{N}))$$

$$\rightarrow (\text{reduce}(\text{rational}(v * z, v * w)) = \text{reduce}(\text{rational}(z, w)))$$

THEOREM: rplus-rzerop-bridge2

$$(v \not\approx 0)$$

$$\rightarrow (\text{reduce}(\text{rational}(- (d * v), v * w))$$

$$= \text{reduce}(\text{rational}(- d, w)))$$

THEOREM: rplus-rzerop

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

THEOREM: eval\$rplus

$$(\text{car}(x) = \text{'rplus})$$

$$\rightarrow (\text{eval}\$(\mathbf{t}, x, a) = \text{rplus}(\text{eval}\$(\mathbf{t}, \text{cadr}(x), a), \text{eval}\$(\mathbf{t}, \text{caddr}(x), a)))$$

THEOREM: eval\$-reduce

$$(\text{car}(x) = \text{'reduce}) \rightarrow (\text{eval}\$(\mathbf{t}, x, a) = \text{reduce}(\text{eval}\$(\mathbf{t}, \text{cadr}(x), a)))$$

THEOREM: reduce-eval\$-rplus-tree
 $\text{reduce}(\text{eval\$}(\mathbf{t}, \text{rplus-tree}(y), a)) = \text{eval\$}(\mathbf{t}, \text{rplus-tree}(y), a)$

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

THEOREM: member-append
 $(a \in \text{append}(x, y)) = ((a \in x) \vee (a \in y))$

THEOREM: delete-append
 $\text{delete}(x, \text{append}(a, b))$
 $= \text{if } x \in a \text{ then } \text{append}(\text{delete}(x, a), b)$
 $\quad \text{else } \text{append}(a, \text{delete}(x, b)) \text{ endif}$

THEOREM: member-subbagp
 $((a \in x) \wedge \text{subbagp}(x, y)) \rightarrow (a \in y)$

DEFINITION:
 $\text{badguy}(x, y)$
 $= \text{if listp}(x)$
 $\quad \text{then if } \text{car}(x) \in y \text{ then } \text{badguy}(\text{cdr}(x), \text{delete}(\text{car}(x), y))$
 $\quad \text{else } \text{car}(x) \text{ endif}$
 $\text{else } 0 \text{ endif}$

THEOREM: member-occur
 $(a \in x) = (0 < \text{occurrences}(a, x))$

THEOREM: occurrences-delete2
 $\text{occurrences}(a, \text{delete}(b, x))$
 $= \text{if } a = b \text{ then } \text{occurrences}(a, x) - 1$
 $\quad \text{else } \text{occurrences}(a, x) \text{ endif}$

THEOREM: subbagp-wit-lemma
 $\text{subbagp}(x, y)$
 $= (\text{occurrences}(\text{badguy}(x, y), y) \not\leq \text{occurrences}(\text{badguy}(x, y), x))$

THEOREM: occurrences-append
 $\text{occurrences}(a, \text{append}(x, y)) = (\text{occurrences}(a, x) + \text{occurrences}(a, y))$

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

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

$$\text{subbagp}(x, y) \rightarrow \text{subbagp}(\text{delete}(a, x), \text{delete}(a, y))$$

THEOREM: subbagp-delete-same-means

$$((a \in y) \wedge \text{subbagp}(\text{delete}(a, x), \text{delete}(a, y))) \rightarrow \text{subbagp}(x, y)$$

THEOREM: subbagp-delete-car

$$\text{subbagp}(x, y) \rightarrow \text{subbagp}(\text{delete}(\text{car}(y), x), \text{cdr}(y))$$

THEOREM: subbagp-delete-car2

$$\text{subbagp}(x, y) \rightarrow \text{subbagp}(\text{cdr}(x), \text{delete}(\text{car}(x), y))$$

THEOREM: subbagp-permutation

$$(\text{subbagp}(x, y) \wedge \text{subbagp}(y, x)) \rightarrow \text{permutation}(x, y)$$

THEOREM: permutation-a-b

$$\text{permutation}(\text{append}(a, b), \text{append}(b, a))$$

THEOREM: not-subbagp-not-permutation

$$(\neg \text{subbagp}(x, y)) \rightarrow ((\neg \text{permutation}(x, y)) \wedge (\neg \text{permutation}(y, x)))$$

THEOREM: permutation-as-subbagp-helper

$$\text{permutation}(x, y) \leftrightarrow (\text{subbagp}(x, y) \wedge \text{subbagp}(y, x))$$

THEOREM: permutation-as-subbagp

$$\text{permutation}(x, y) = (\text{subbagp}(x, y) \wedge \text{subbagp}(y, x))$$

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-nec

$$\text{subbagp}(x, y) \rightarrow (\text{occurrences}(a, y) \not\prec \text{occurrences}(a, x))$$

THEOREM: subbagp-transitive

$$(\text{subbagp}(x, y) \wedge \text{subbagp}(y, z)) \rightarrow \text{subbagp}(x, z)$$

THEOREM: not-member-make-negs-fact

$$(x \notin y) \rightarrow (\text{delete}(\text{list}(' \mathbf{rneg}, x), \text{make-negs}(y)) = \text{make-negs}(y))$$

THEOREM: make-negs-delete

$$\text{make-negs}(\text{delete}(a, x)) = \text{delete}(\text{list}(' \mathbf{rneg}, a), \text{make-negs}(x))$$

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

$$\begin{aligned} &(\text{bagint}(a, c) = \mathbf{nil}) \\ \rightarrow &(\text{append}(c, \text{bagdiff}(b, a)) = \text{bagdiff}(\text{append}(c, b), a)) \end{aligned}$$

THEOREM: bagdiff-bagdiff

$$\text{bagdiff}(\text{bagdiff}(a, b), c) = \text{bagdiff}(a, \text{append}(b, c))$$

THEOREM: member-rneg-make-negs

$$(\text{list}(\mathbf{'rneg}, x) \in \text{make-negs}(y)) = (x \in y)$$

THEOREM: subbagp-make-negs

$$\text{subbagp}(\text{make-negs}(x), \text{make-negs}(y)) = \text{subbagp}(x, y)$$

THEOREM: rdifference-reduce

$$\begin{aligned} &(\text{rdifference}(\text{reduce}(x), y) = \text{rdifference}(x, y)) \\ \wedge &(\text{rdifference}(x, \text{reduce}(y)) = \text{rdifference}(x, y)) \end{aligned}$$

THEOREM: requal-simple-rplus-x-simple-rneg-x

$$\text{requal}(\text{simple-rplus}(x, \text{simple-rneg}(x)), \text{rational}(0, 1))$$

THEOREM: requal-rplus-x-simple-rneg-x

$$\text{requal}(\text{rplus}(x, \text{simple-rneg}(x)), \text{rational}(0, 1))$$

THEOREM: rplus-x-rneg-x

$$\text{rplus}(x, \text{rneg}(x)) = \text{rational}(0, 1)$$

THEOREM: rdifference-x-x

$$\text{rdifference}(x, x) = \text{rational}(0, 1)$$

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

$$\begin{aligned} &(\text{rplus}(a, \text{rdifference}(b, a)) = \text{reduce}(b)) \\ \wedge &(\text{rplus}(\text{rdifference}(b, a), a) = \text{reduce}(b)) \end{aligned}$$

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
 $(\text{fix}(b) = \text{fix}(d))$
 $\rightarrow (((a - b) = (c - d))$
 $\quad = (((b \not\prec a) \wedge (b \not\prec c)) \vee (a = c)))$

THEOREM: equal-difference-rewrite2
 $(\text{fix}(a) = \text{fix}(c))$
 $\rightarrow (((a - b) = (c - d))$
 $\quad = (((b \not\prec a) \wedge (d \not\prec c)) \vee (\text{fix}(b) = \text{fix}(d))))$

THEOREM: equal-times-bridge5
 $((b * a) = (c * d)) \wedge ((x * a) = (c * y)) \wedge (a \not\approx 0)$
 $\rightarrow ((b * y) = (x * d)) = \mathbf{t}$

THEOREM: equal-plus-difference-rewrite
 $(\text{fix}(a) = \text{fix}(c))$
 $\rightarrow (((a + b) = (c - d)) = ((b \simeq 0) \wedge ((a \simeq 0) \vee (d \simeq 0))))$
 $\quad \wedge (((b + a) = (c - d))$
 $\quad \quad = ((b \simeq 0) \wedge ((a \simeq 0) \vee (d \simeq 0))))$
 $\quad \wedge (((c - d) = (a + b))$
 $\quad \quad = ((b \simeq 0) \wedge ((a \simeq 0) \vee (d \simeq 0))))$
 $\quad \wedge (((c - d) = (b + a))$
 $\quad \quad = ((b \simeq 0) \wedge ((a \simeq 0) \vee (d \simeq 0))))$

THEOREM: lessp-times-bridge1
 $((c * v) < (x1 * z1))$
 $\quad \wedge ((w * x1) = (c * d))$
 $\quad \wedge (c \not\approx 0)$
 $\quad \wedge (d \not\approx 0)$
 $\rightarrow ((v * w) < (d * z1))$

THEOREM: equal-difference
 $((a = (a - b)) = ((a \in \mathbf{N}) \wedge ((a \simeq 0) \vee (b \simeq 0))))$
 $\quad \wedge (((a - b) = a) = ((a \in \mathbf{N}) \wedge ((a \simeq 0) \vee (b \simeq 0))))$

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

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

THEOREM: permutation-does-not-effect-rplus
 $\text{permutation}(x, y)$
 $\rightarrow (\text{eval\$}(\mathbf{t}, \text{rplus-tree}(x), a) = \text{eval\$}(\mathbf{t}, \text{rplus-tree}(y), a))$

THEOREM: eval\$-rplus-tree-zero
 $\text{eval\$}(\mathbf{t}, \text{rplus-tree}(\text{append}(\text{make-negs}(x), x)), a) = \text{rational}(0, 1)$

THEOREM: cancel-zero-fact
 $(\text{eval\$}(\mathbf{t}, \text{rplus-tree}(z), a) = \text{rational}(0, 1))$
 $\rightarrow (\text{eval\$}(\mathbf{t}, \text{rplus-tree}(\text{append}(x, z)), a) = \text{eval\$}(\mathbf{t}, \text{rplus-tree}(x), a))$

THEOREM: member-car-x-x
 $(\text{car}(x) \in x) = \text{listp}(x)$

THEOREM: member-bagdiff-append
 $(e \in \text{bagdiff}(\text{append}(x, z), z)) = (e \in x)$

THEOREM: permutation-transitive
 $(\text{permutation}(x, y) \wedge \text{permutation}(y, z)) \rightarrow \text{permutation}(x, z)$

DEFINITION:
 $\text{last-cdr}(x)$
 $= \text{if } \text{listp}(x) \text{ then } \text{last-cdr}(\text{cdr}(x))$
 $\text{else } x \text{ endif}$

THEOREM: bagdiff-x-x
 $\text{bagdiff}(x, x) = \text{last-cdr}(x)$

THEOREM: bagdiff-append-arg1
 $\text{bagdiff}(\text{append}(z, x), z) = x$

THEOREM: bagdiff-cons-z-z
 $\text{bagdiff}(\text{cons}(x, z), z) = \text{cons}(x, \text{last-cdr}(z))$

THEOREM: bagdiff-not-listp
 $(\neg \text{listp}(x)) \rightarrow (\text{bagdiff}(x, z) = x)$

THEOREM: bagdiff-car-in
 $(\text{listp}(x) \wedge (\text{car}(x) \in z))$
 $\rightarrow (\text{bagdiff}(x, z) = \text{bagdiff}(\text{cdr}(x), \text{delete}(\text{car}(x), z)))$

THEOREM: last-cdr-delete

$$\text{last-cdr}(\text{delete}(e, x)) = \text{last-cdr}(x)$$

THEOREM: member-subbagp2

$$(\text{subbagp}(x, y) \wedge (e \in x)) \rightarrow (e \in y)$$

THEOREM: member-subbagp-delete

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

THEOREM: subbagp-bagdiff

$$\text{subbagp}(x, y) \rightarrow \text{subbagp}(\text{bagdiff}(x, z), \text{bagdiff}(y, z))$$

THEOREM: permutation-bagdiff

$$\text{permutation}(x, y) \rightarrow \text{permutation}(\text{bagdiff}(x, z), \text{bagdiff}(y, z))$$

THEOREM: permutation-append-arg1-arg2-bridge

$$\text{permutation}(\text{bagdiff}(\text{append}(x, z), z), \text{bagdiff}(\text{append}(z, x), z))$$

THEOREM: subbagp-transitive-bridge-helper

$$\begin{aligned} &(\text{subbagp}(y, z) \wedge \text{subbagp}(z, y)) \\ \rightarrow &(((\text{subbagp}(y, x) \leftrightarrow \text{subbagp}(z, x)) = \mathbf{t}) \\ &\wedge ((\text{subbagp}(x, y) \leftrightarrow \text{subbagp}(x, z)) = \mathbf{t})) \end{aligned}$$

THEOREM: subbagp-transitive-bridge

$$\begin{aligned} &(\text{subbagp}(y, z) \wedge \text{subbagp}(z, y)) \\ \rightarrow &(((\text{subbagp}(y, x) = \text{subbagp}(z, x)) = \mathbf{t}) \\ &\wedge ((\text{subbagp}(x, y) = \text{subbagp}(x, z)) = \mathbf{t})) \end{aligned}$$

THEOREM: equal-permutation

$$\begin{aligned} &(\text{permutation}(y, z) \rightarrow ((\text{permutation}(x, y) = \text{permutation}(x, z)) = \mathbf{t})) \\ \wedge &(\text{permutation}(y, z) \rightarrow ((\text{permutation}(y, x) = \text{permutation}(z, x)) = \mathbf{t})) \end{aligned}$$

THEOREM: permutation-bagdiff-append-helper

$$\begin{aligned} &(\text{permutation}(\text{bagdiff}(\text{append}(z, x), x), y) \\ &= \text{permutation}(\text{bagdiff}(\text{append}(x, z), x), y)) \\ \wedge &(\text{permutation}(y, \text{bagdiff}(\text{append}(z, x), x)) \\ &= \text{permutation}(y, \text{bagdiff}(\text{append}(x, z), x))) \end{aligned}$$

THEOREM: permutation-bagdiff-append

$$\text{permutation}(x, \text{bagdiff}(\text{append}(x, z), z))$$

THEOREM: cancel-zero-fact-bridge

$$\begin{aligned} &(\text{subbagp}(z, x) \wedge (\text{eval}(\mathbf{t}, \text{rplus-tree}(z), a) = \text{rational}(0, 1))) \\ \rightarrow &(\text{eval}(\mathbf{t}, \text{rplus-tree}(\text{bagdiff}(x, z)), a) = \text{eval}(\mathbf{t}, \text{rplus-tree}(x), a)) \end{aligned}$$

THEOREM: rnegs-cancel-list

$$\text{eval}(\mathbf{t}, \text{rplus-tree}(\text{append}(x, \text{make-negs}(x))), a) = \text{rational}(0, 1)$$

THEOREM: subbagp-append-simplify1
 $\text{subbagp}(a, x) \rightarrow \text{subbagp}(a, \text{append}(x, y))$

THEOREM: subbagp-permutation-equiv
 $(\text{permutation}(x, y) \wedge \text{subbagp}(a, x)) \rightarrow \text{subbagp}(a, y)$

THEOREM: subbagp-append-simplify2
 $\text{subbagp}(a, x) \rightarrow \text{subbagp}(a, \text{append}(y, x))$

THEOREM: subbagp-append-bridge
 $(\text{subbagp}(x, b) \wedge \text{subbagp}(y, a)) \rightarrow \text{subbagp}(\text{append}(x, y), \text{append}(b, a))$

THEOREM: subbagp-append-bridge2
 $(\text{subbagp}(x, b) \wedge \text{subbagp}(y, a)) \rightarrow \text{subbagp}(\text{append}(x, y), \text{append}(a, b))$

THEOREM: cancelling-from-rplus
 $(\text{subbagp}(z, x) \wedge \text{subbagp}(z, y) \wedge (\text{bagint}(\text{make-negs}(z), x) = \mathbf{nil}))$
 $\rightarrow (\text{eval}\$ (\mathbf{t}, \text{rplus-tree}(\text{append}(\text{bagdiff}(x, z), \text{make-negs}(\text{bagdiff}(y, z)))), a)$
 $= \text{eval}\$ (\mathbf{t}, \text{rplus-tree}(\text{append}(x, \text{make-negs}(y))), a)$

THEOREM: eval\$rplus-tree-rplus-fringe
 $(\text{car}(x) = \mathbf{rplus})$
 $\rightarrow (\text{eval}\$ (\mathbf{t}, \text{rplus-tree}(\text{rplus-fringe}(x)), a) = \text{eval}\$ (\mathbf{t}, x, a))$

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

THEOREM: rnegs-not-car-split-by-parity
 $((\text{car}(x) = \mathbf{rneg}) \wedge (\text{caadr}(x) \neq \mathbf{rneg}) \wedge (\text{cddr}(x) = \mathbf{nil}))$
 $\rightarrow (x \notin \text{car}(\text{split-by-parity}(y)))$

THEOREM: permutation-append-split-by-parity-bridge
 $\text{permutation}(\text{append}(\text{car}(\text{split-by-parity}(x)),$
 $\text{make-negs}(\text{cdr}(\text{split-by-parity}(x)))),$
 $x)$

THEOREM: member-cdr-split-by-parity
 $(e \in \text{cdr}(\text{split-by-parity}(x))) \rightarrow (\text{car}(e) \neq \mathbf{rneg})$

DEFINITION:
 $\text{all-rnegs}(x)$
 $= \mathbf{if} \text{ listp}(x)$
 $\quad \mathbf{then} (\text{caar}(x) = \mathbf{rneg})$
 $\quad \quad \wedge (\text{caadar}(x) \neq \mathbf{rneg})$
 $\quad \quad \wedge (\text{cddar}(x) = \mathbf{nil})$
 $\quad \quad \wedge \text{all-rnegs}(\text{cdr}(x))$
 $\quad \mathbf{else} \mathbf{t} \mathbf{endif}$

THEOREM: all-rnegs-make-negs-bagint-fact
 $(\text{all-rnegs}(\text{make-negs}(\text{bagint}(z, w))) \wedge (v \in w))$
 $\rightarrow \text{all-rnegs}(\text{make-negs}(\text{bagint}(z, \text{delete}(v, w))))$

THEOREM: all-rnegs-make-negs-bagint
 $\text{all-rnegs}(\text{make-negs}(\text{bagint}(\text{car}(\text{split-by-parity}(x)), \text{cdr}(\text{split-by-parity}(y))))))$

THEOREM: bagint-all-rnegs-car-split-by-parity
 $\text{all-rnegs}(x) \rightarrow (\text{bagint}(x, \text{car}(\text{split-by-parity}(y))) = \mathbf{nil})$

THEOREM: bagint-make-negs-split-by-parity
 $\text{bagint}(\text{make-negs}(\text{bagint}(\text{car}(\text{split-by-parity}(y)), \text{cdr}(\text{split-by-parity}(y))))),$
 $\text{car}(\text{split-by-parity}(y)))$
 $= \mathbf{nil}$

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

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\$r-plus-tree; name this event 'reduce-eval\$r-plus-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\$ ($\mathbf{t}$, x , a) = eval\$ ($\mathbf{t}$, 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

- addition, 15
- all-rnegs, 88, 89
- all-rnegs-make-negs-bagint, 89
- all-rnegs-make-negs-bagint-fact, 89
 - off, 90
- all-rnegs-make-negs-bagint-off, 90
- and-not-zero-tree, 45, 47–49
- append-bagdiff-arg1, 84
- append-bagdiff-arg1-off, 93
- append-bagdiff-arg2, 84
- append-bagdiff-arg2-off, 93
- arithmetic, 30
- associativity-of-gcd, 30
- associativity-of-gcd-zero-case, 30
- associativity-of-iplus, 33
- associativity-of-itimes, 37
- associativity-of-plus, 5
- associativity-of-rplus, 65
- associativity-of-rplus-off, 72
- associativity-of-rtimes, 67
- associativity-of-rtimes-off, 70
- associativity-of-simple-rtimes, 66
- associativity-of-simple-rtimes-off, 70
 - when-not-rzerop, 66
 - when-not-rzerop-off, 71
- associativity-of-times, 16
- badguy, 82
- bagdiff, 1–3, 8–10, 12, 35–37, 46–50, 84, 86–88, 95
- bagdiff-append-arg1, 86
- bagdiff-append-arg1-off, 92
- bagdiff-bagdiff, 84
- bagdiff-bagdiff-off, 93
- bagdiff-car-in, 86
- bagdiff-car-in-off, 91
- bagdiff-cons-z-z, 86
- bagdiff-cons-z-z-off, 91
- bagdiff-delete, 46
- bagdiff-delete-off, 51
- bagdiff-not-listp, 86
- bagdiff-not-listp-off, 91
- bagdiff-x-x, 86
- bagdiff-x-x-off, 92
- bagint, 1–3, 5, 9, 10, 12, 35–37, 47–50, 84, 88, 89, 95
- bagint-all-rnegs-car-split-by-p
 - arity, 89
 - arity-off, 90
- bagint-make-negs-split-by-parit
 - y, 89
 - y-off, 90
- bridge-to-subbagp-implies-plus-t
 - ree-greatereqp, 9
 - ree-greatereqp-off, 13
- cadr-eval\$list, 9
- cadr-eval\$list-off, 13
- cancel-difference-plus, 10, 11
- cancel-difference-plus-off, 13
- cancel-equal-plus, 9, 10
- cancel-equal-plus-off, 13
- cancel-equal-times, 49, 50
- cancel-equal-times-preserves-inequality, 50
 - uality-bridge, 50
 - uality-bridge-off, 52
 - uality-off, 52
- cancel-idifference, 36, 37
- cancel-idifference-off, 43
- cancel-iplus, 35, 36
- cancel-iplus-off, 44
- cancel-lessp-plus, 12
- cancel-lessp-plus-off, 12
- cancel-lessp-times, 47, 48
- cancel-quotient-times, 48, 49
- cancel-rplus, 95
- cancel-rplus-off, 95
- cancel-zero-fact, 86
- cancel-zero-fact-bridge, 87
- cancel-zero-fact-bridge-off, 89

cancel-zero-fact-off, 89
cancelling-from-rplus, 88
cancelling-from-rplus-off, 89
common-divisor-divides-gcd, 30
commutativity-of-gcd, 29
commutativity-of-iplus, 33
commutativity-of-itimes, 37
commutativity-of-permutation, 3
commutativity-of-plus, 5
commutativity-of-requal, 61
commutativity-of-requal-off, 75
commutativity-of-rplus, 62
commutativity-of-rplus-off, 75
commutativity-of-rtimes, 66
commutativity-of-rtimes-off, 71
commutativity-of-simple-rplus, 68
commutativity-of-simple-rplus-off, 69
commutativity-of-simple-rtimes, 66
commutativity-of-simple-rtimes-off, 71
commutativity-of-times, 15
commutativity2-of-gcd, 30
commutativity2-of-gcd-zero-case, 30
commutativity2-of-iplus, 33
commutativity2-of-itimes, 37
commutativity2-of-plus, 5
commutativity2-of-rplus, 65
commutativity2-of-rplus-off, 72
commutativity2-of-rtimes, 89
commutativity2-of-times, 15
correctness-of-cancel-difference-plus, 11
correctness-of-cancel-equal-plus, 10
correctness-of-cancel-equal-time, 50
correctness-of-cancel-idifference, 37
correctness-of-cancel-iplus, 36
correctness-of-cancel-lessplu, 12
correctness-of-cancel-lessplu-time, 48
correctness-of-cancel-quotient-times, 49
correctness-of-cancel-rplus, 95
correctness-of-cancel-rplus-off, 95
delete, 1–4, 8, 10, 12, 35–37, 46, 47, 82, 83, 86, 87, 89
delete-append, 82
delete-append-off, 95
delete-delete, 2
delete-non-member, 2
denominator, 53, 54, 56, 61–63
diff-add1-arg2, 11
diff-add1-arg2-casesplit, 11
diff-diff-arg1, 11
diff-diff-arg2, 11
diff-diff-arg2-casesplit, 11
diff-diff-diff, 11
diff-diff-diff-casesplit, 11
diff-sub1-arg2, 11
diff-sub1-arg2-casesplit, 11
diff-x-x, 12
difference-1, 7
difference-1-off, 14
difference-2, 7
difference-2-off, 14
difference-add1-plus-cancellation-on, 7
difference-add1-plus-cancellation-on-off, 14
difference-cancellation-0, 7
difference-cancellation-0-off, 14
difference-cancellation-1, 7
difference-cancellation-1-off, 14
difference-difference-arg1, 7
difference-difference-arg1-off, 14
difference-difference-arg2, 7
difference-difference-arg2-off, 14
difference-difference-difference, 8
difference-difference-difference-off, 14
difference-elim, 6
difference-leq-arg1, 6
difference-plus, 7
difference-plus-cancellation1, 7
difference-plus-cancellation1-off, 14

- ff, 14
- difference-plus-cancellation2, 7
- difference-plus-cancellation2-o
 - ff, 14
- difference-plus-cancellation3, 8
- difference-plus-cancellation3-o
 - ff, 14
- difference-plus-off, 14
- difference-sub1-arg2, 7
- difference-sub1-arg2-off, 14
- difference-x-x, 7
- difference-x-x-off, 15
- distributivity-of-times-over-gc
 - d, 30
 - d-proof, 30
- divides-each-equality, 57
- divides-each-equality-off, 77
- divides-product-reduction, 58
- divides-product-reduction-off, 76
- division-theorem, 39
- division-theorem-for-truncate-t
 - o-neginf, 40
 - o-neginf-part1, 39
 - o-neginf-part2, 39
 - o-neginf-part3, 40
 - o-zero, 41
 - o-zero-part1, 40
 - o-zero-part2, 40
 - o-zero-part3, 40
- division-theorem-part1, 38
- division-theorem-part2, 38
- division-theorem-part3, 38
- double-log-induction, 27
- double-number-induction, 22
- double-remainder-induction, 20
- dpr-hack1, 58
- dpr-hack1-off, 59
- dpr-hack1-off1, 77
- dpr-hack2, 58
- dpr-hack2-off, 59
- dpr-hack2-off1, 77
- dpr-hack3, 58
- dpr-hack3-off, 59
- dpr-hack3-off1, 77
- dpr-hack4, 58
- dpr-hack4-off, 59
- dpr-hack4-off1, 77
- dpr-hack5, 58
- dpr-hack5-off, 59
- dpr-hack5-off1, 76
- equal-difference, 85
- equal-difference-0, 7
- equal-difference-hack1, 84
- equal-difference-hack1-off, 93
- equal-difference-hack2, 85
- equal-difference-hack2-off, 93
- equal-difference-off, 92
- equal-difference-rewrite, 85
- equal-difference-rewrite-off, 92
- equal-difference-rewrite2, 85
- equal-difference-rewrite2-off, 92
- equal-exp-0, 27
- equal-exp-1, 27
- equal-gcd-0, 29
- equal-idifference-0, 34
- equal-ineg-ineg, 55
- equal-ineg-ineg-off, 79
- equal-imes-0, 38
- equal-imes-1, 38
- equal-imes-minus-1, 38
- equal-log-0, 27
- equal-occurrences-zero, 2
- equal-permutation, 87
- equal-permutation-off, 91
- equal-plus-0, 5
- equal-plus-1, 6
- equal-plus-difference-rewrite, 85
- equal-plus-difference-rewrite-o
 - ff, 92
- equal-quotient-0, 23
- equal-remainder-difference-0, 20
- equal-remainder-difference-0-of
 - f, 22
- equal-remainder-plus-0, 19
- equal-remainder-plus-0-proof, 19
- equal-remainder-plus-remainder, 19
- equal-remainder-plus-remainder-

- off, 22
- equal-remainder-plus-remainder-p
 - roof, 19
- equal-requal-rewrite, 64
- equal-requal-rewrite-off, 65
- equal-requal-rewrite-off1, 72
- equal-rplus-rdifference-hack, 85
- equal-rplus-rdifference-hack-of
 - f, 92
- equal-rplus-x-x-rewrite, 85
- equal-rplus-x-x-rewrite-off, 92
- equal-sub1-times-0, 18
- equal-times-0, 16
- equal-times-1, 16
- equal-times-arg1, 47
- equal-times-arg1-off, 51
- equal-times-bridge, 47
- equal-times-bridge-off, 51
- equal-times-bridge1, 64
- equal-times-bridge1-off, 73
- equal-times-bridge2, 64
- equal-times-bridge2-off, 73
- equal-times-bridge3, 64
- equal-times-bridge3-off, 73
- equal-times-bridge4, 64
- equal-times-bridge4-off, 73
- equal-times-bridge5, 85
- equal-times-bridge5-off, 92
- equal-times-gcd-bridge1, 61
- equal-times-gcd-bridge1-off, 76
- equal-times-times-quotient-arg2, 60
 - off, 76
- eval\$-equal, 46
- eval\$-equal-off, 50
- eval\$-equal-tree-bagdiff, 49
 - off, 52
- eval\$-if, 46
- eval\$-if-off, 51
- eval\$-if2, 46
- eval\$-if2-off, 51
- eval\$-iplus-tree-append, 36
- eval\$-iplus-tree-append-inducti
 - on, 36
- eval\$-iplus-tree-append-off, 43
- eval\$-lessp, 46
- eval\$-lessp-off, 50
- eval\$-lessp-times-tree-bagdiff, 48
 - off, 52
- eval\$-lessp2, 46
- eval\$-lessp2-off, 50
- eval\$-or, 46
- eval\$-or-off, 50
- eval\$-or2, 46
- eval\$-or2-off, 50
- eval\$-plus-tree-append, 9
- eval\$-plus-tree-append-off, 13
- eval\$-quote, 9
- eval\$-quote-off, 13
- eval\$-quotient, 46
- eval\$-quotient-off, 51
- eval\$-quotient-times-tree-bagdi
 - ff, 49
 - ff-off, 52
- eval\$-quotient2, 46
- eval\$-quotient2-off, 51
- eval\$-reduce, 81
- eval\$-reduce-off, 95
- eval\$-rplus, 81
- eval\$-rplus-off, 95
- eval\$-rplus-tree-delete, 86
- eval\$-rplus-tree-delete-off, 89
- eval\$-rplus-tree-rplus-fringe, 88
- eval\$-rplus-tree-rplus-fringe-
 - off, 89
- eval\$-rplus-tree-zero, 86
- eval\$-rplus-tree-zero-off, 89
- eval\$-times, 45
- eval\$-times-member, 47
- eval\$-times-member-off, 51
- eval\$-times-off, 50
- eval\$-times2, 45
- eval\$-times2-off, 50
- exp, 15, 21, 22, 25–28
- exp-0-arg1, 26
- exp-0-arg2, 27
- exp-1-arg1, 26
- exp-add1, 26
- exp-difference, 27

- exp-exp, 27
- exp-plus, 26
- exp-times, 27
- exp-zero, 26
- exponentiation, 27

- fix-int, 31–34, 36–42, 54
- fix-int-fix-int, 32
- fix-int-iabs, 32
- fix-int-idifference, 32
- fix-int-idiv, 42
- fix-int-imod, 42
- fix-int-ineg, 32
- fix-int-iplus, 32
- fix-int-iquo, 42
- fix-int-iquotient, 42
- fix-int-irem, 42
- fix-int-iremainder, 42
- fix-int-itypes, 32
- fix-int-off, 44
- fix-int-off1, 59
- fix-int-on, 57
- fix-int-on-integers, 54
- fix-int-on-integers-off, 79
- fix-rational, 53–56, 62–64, 67
- fix-rational-fix-rational, 56
- fix-rational-fix-rational-off, 78
- fix-rational-of-rationalp, 63
- fix-rational-of-rationalp-off, 73
- fix-rational-off, 80
- fix-rational-rdifference, 56
- fix-rational-rdifference-off, 78
- fix-rational-reduce, 56
- fix-rational-reduce-off, 78
- fix-rational-rmagnitude, 56
- fix-rational-rmagnitude-off, 78
- fix-rational-rneg, 56
- fix-rational-rneg-off, 78
- fix-rational-rplus, 56
- fix-rational-rplus-off, 78
- fix-rational-rquotient, 56
- fix-rational-rquotient-off, 78
- fix-rational-rtimes, 56
- fix-rational-rtimes-off, 78

- fix-rational-simple-rplus, 64
- fix-rational-simple-rplus-off, 73
- fix-rational-simple-rtimes, 67
- fix-rational-simple-rtimes-off, 70

- gcd, 28–30, 53, 56–58, 60, 61
- gcd-0, 29
- gcd-1, 29
- gcd-factors, 57, 58
- gcd-factors-gives-linear-combination, 57
- gcd-factors-gives-linear-combination-off, 59
- gcd-factors-gives-linear-combination-off1, 77
- gcd-factors-gives-linear-combination-rewrite, 58
- gcd-factors-gives-linear-combination-rewrite-off, 59
- gcd-factors-gives-linear-combination-rewrite-off1, 77
- gcd-factors-off, 59
- gcd-factors-off1, 77
- gcd-idempotence, 30
- gcd-is-the-greatest, 30
- gcd-plus, 29
- gcd-plus-instance, 29
- gcd-plus-instance-temp, 29
- gcd-plus-instance-temp-proof, 29
- gcd-plus-proof, 29
- gcd-quotient-quotient, 57
- gcd-quotient-quotient-off, 77
- gcd-remainder-fact1, 56
- gcd-remainder-fact1-off, 77
- gcd-remainder-fact2, 56
- gcd-remainder-fact2-off, 77
- gcd-remainder-times-fact1-proof, 60
- gcd-remainder-times-fact1-proof-off, 76
- gcd-times1, 56
- gcd-times1-induct, 56
- gcd-times1-induct-off, 77
- gcd-times1-off, 77
- gcd-times2, 56
- gcd-times2-off, 77
- gcd-x-x, 30
- gcds, 30

- iabs, 31, 32, 38–41
- iabs-off, 44

- idifference, 31–34, 36–40, 57
- idifference-0, 34
- idifference-cancellation-1, 34
- idifference-fix-int, 34
- idifference-idifference, 34
- idifference-iplus, 34
- idifference-iplus-canonicalizer
 - 1-off, 57
 - 1-on, 59
 - 1, 34
- idifference-off, 44
- idifference-off1, 59
- idifference-on, 57
- idifference-right-id, 34
- idifference-x-x, 34
- idiv, 39–42
- idiv-fix-int1, 42
- idiv-fix-int2, 42
- idiv-imod-uniqueness, 40
- idiv-off, 43
- ileq, 31, 38–41
- ileq-off, 44
- ilessp, 31, 38–41, 54
- ilessp-off, 44
- ilessp-off1, 59
- ilessp-on, 57
- imod, 39–42
- imod-fix-int1, 42
- imod-fix-int2, 42
- imod-off, 43
- ineg, 31–33, 40, 41, 53–55, 62, 64
- ineg-fix-int, 33
- ineg-id, 32
- ineg-idifference, 33
- ineg-ineg, 33
- ineg-iplus, 33
- ineg-off, 44
- ineg-off1, 59
- ineg-on, 57
- integerp, 30–36, 38–41, 53–56, 61, 63
- integerp-eval\$-bridge, 35
- integerp-eval\$-bridge-off, 43
- integerp-eval\$-iplus, 35
- integerp-eval\$-iplus-off, 44
- integerp-eval\$-iplus-tree, 35
- integerp-eval\$-iplus-tree-off, 43
- integerp-fix-int, 32
- integerp-iabs, 32
- integerp-idifference, 32
- integerp-idiv, 41
- integerp-if-negativep-non-zero, 55
- integerp-if-negativep-non-zero-off, 79
- integerp-if-numberp, 55
- integerp-if-numberp-off, 79
- integerp-imod, 41
- integerp-ineg, 32
- integerp-iplus, 32
- integerp-iquo, 41
- integerp-iquotient, 41
- integerp-irem, 41
- integerp-remainder, 41
- integerp-imes, 32
- integerp-minus, 54
- integerp-minus-off, 79
- integerp-off, 44
- integerp-on, 57
- integers, 45
- iplus, 31–34, 36–41, 53, 55, 58, 62
- iplus-0, 33
- iplus-cancellation, 33
- iplus-fix-int1, 33
- iplus-fix-int2, 33
- iplus-fringe, 34–37
- iplus-idifference-arg1, 33
- iplus-idifference-arg2, 34
- iplus-idifference-idifference, 36
- iplus-ineg1, 33
- iplus-ineg2, 33
- iplus-is-plus, 55
- iplus-is-plus-off, 79
- iplus-left-id, 33
- iplus-off, 44
- iplus-off1, 59
- iplus-on, 57
- iplus-right-id, 33
- iplus-tree, 34–37

- iplus-tree-bagdiff, 36
- iplus-tree-bagdiff-off, 43
- iplus-tree-delete, 36
- iplus-tree-delete-off, 43
- iplus-tree-iplus-fringe, 36
- iplus-tree-iplus-fringe-off, 43
- iquo, 40–42
- iquo-fix-int1, 42
- iquo-fix-int2, 42
- iquo-irem-uniqueness, 41
- iquo-off, 43
- iquotient, 38, 39, 41, 42, 58
- iquotient-fix-int1, 41
- iquotient-fix-int2, 41
- iquotient-iremremainder-uniqueness, 39
- iquotient-off, 43
- iquotient-off1, 60
- iquotient-on, 58
- irem, 40–42
- irem-fix-int1, 42
- irem-fix-int2, 42
- irem-off, 43
- iremremainder, 38, 39, 41, 42, 58
- iremremainder-fix-int1, 41
- iremremainder-fix-int2, 42
- iremremainder-off, 43
- iremremainder-off1, 60
- iremremainder-on, 58
- itimes, 32, 37–41, 53–55, 57, 58, 61, 62
- itimes-1-arg1, 38
- itimes-distributes-over-idifference, 38
- itimes-distributes-over-idifference-proof, 37
- itimes-distributes-over-iplus, 37
- itimes-distributes-over-iplus-proof, 37
- itimes-fix-int1, 37
- itimes-fix-int2, 37
- itimes-ineg-arg1, 54
- itimes-ineg-arg1-off, 79
- itimes-ineg-arg2, 55
- itimes-ineg-arg2-off, 79
- itimes-is-times, 55
- itimes-is-times-off, 79
- itimes-negativep-arg1, 55
- itimes-negativep-arg1-off, 79
- itimes-negativep-arg2, 55
- itimes-negativep-arg2-off, 79
- itimes-off, 44
- itimes-off1, 59
- itimes-on, 57
- itimes-zero1, 37
- itimes-zero2, 37
- izerop, 31, 37–40
- izerop-off, 44
- izerop-off1, 60
- izerop-on, 57
- last-cdr, 86, 87
- last-cdr-delete, 87
- last-cdr-delete-off, 91
- leq-log-log, 28
- leq-quotient, 25
- lessp-1-times, 18
- lessp-difference, 12
- lessp-difference-cancellation, 12
- lessp-difference-cancellation-off, 12
- lessp-gcd, 29
- lessp-plus-fact, 21
- lessp-plus-fact-off, 22
- lessp-plus-times-proof, 17
- lessp-plus-times1, 17
- lessp-plus-times2, 17
- lessp-quotient, 26
- lessp-remainder, 18
- lessp-times-arg1, 46
- lessp-times-arg1-off, 51
- lessp-times-bridge1, 85
- lessp-times-bridge1-off, 92
- lessp-times-cancellation-proof, 17
- lessp-times-cancellation1, 17
- lessp-times-cancellation2, 17
- lessp-times1, 16
- lessp-times1-proof, 16
- lessp-times2, 16
- lessp-times2-proof, 16

- lessp-times3, 17
- lessp-times3-proof1, 16
- lessp-times3-proof2, 17
- listp-delete, 2
- listp-eval\$, 9
- listp-eval\$-off, 13
- log, 27, 28
- log-0, 27
- log-1, 27
- log-exp, 28
- log-quotient, 28
- log-quotient-exp, 28
- log-quotient-times, 28
- log-quotient-times-proof, 28
- log-times, 28
- log-times-exp, 28
- log-times-exp-proof, 28
- log-times-proof, 28
- logs, 28

- make-negs, 81, 83, 84, 86–89, 95
- make-negs-bagdiff, 84
- make-negs-bagdiff-off, 94
- make-negs-delete, 83
- make-negs-delete-off, 94
- means-rationalp, 63
- means-rationalp-off, 74
- member-append, 82
- member-append-off, 95
- member-bagdiff, 3
- member-bagdiff-append, 86
- member-bagdiff-append-off, 92
- member-bagint, 3
- member-car-x-x, 86
- member-car-x-x-off, 92
- member-cdr-split-by-parity, 88
- member-cdr-split-by-parity-off, 90
- member-cons, 3
- member-delete, 3
- member-delete-implies-membership, 3
- member-implies-numberp, 9
- member-implies-numberp-off, 13
- member-implies-plus-tree-greate

- reqp, 8
- reqp-off, 13
- member-non-list, 2
- member-occur, 82
- member-occur-off, 82
- member-rneg-make-negs, 84
- member-rneg-make-negs-off, 93
- member-subbagp, 82
- member-subbagp-delete, 87
- member-subbagp-delete-off, 91
- member-subbagp-off, 94
- member-subbagp2, 87
- member-subbagp2-off, 91
- multiplication, 18

- negative-guts-ineg, 64
- negative-guts-ineg-off, 73
- negativep-if-integerp-and-not-n
- umberp, 61
- umberp-off, 61
- umberp-off1, 75
- negativep-ineg, 62
- negativep-ineg-off, 74
- not-integerp-implies-not-equal-ip
- lus, 36
- lus-off, 43
- not-member-make-negs-fact, 83
- not-member-make-negs-fact-off, 94
- not-permutation, 3
- not-subbagp-not-permutation, 83
- not-subbagp-not-permutation-off, 94
- nrational-rplus-arg1, 63
- nrational-rplus-arg1-off, 74
- nrational-rplus-arg2, 63
- nrational-rplus-arg2-off, 74
- nrational-simple-rplus-arg1, 63
- nrational-simple-rplus-arg1-off, 74
- nrational-simple-rplus-arg2, 63
- nrational-simple-rplus-arg2-off, 73
- numberp-eval\$-bridge, 8
- numberp-eval\$-bridge-off, 13
- numberp-eval\$-plus, 8
- numberp-eval\$-plus-off, 14
- numberp-eval\$-plus-tree, 8

- numberp-eval\$-plus-tree-off, 13
- numberp-eval\$-times, 45
- numberp-eval\$-times-off, 50
- numberp-eval\$-times-tree, 46
- numberp-eval\$-times-tree-off, 51
- numberp-if-integerp-and-not-neg
 - ativep, 61
 - ativep-off, 61
 - ativep-off1, 75
- numberp-ineg, 62
- numberp-ineg-off, 74
- numberp-numerator-reduce, 62
- numberp-numerator-reduce-off, 74
- numerator, 53, 54, 56, 61–63, 66, 67
- numerator-type, 67
- numerator-type-off, 70
- numerator-zero-rzerop-bridge, 66
- numerator-zero-rzerop-bridge-of
 - f, 71
- occurrences, 1–3, 82, 83
- occurrences-append, 82
- occurrences-append-off, 94
- occurrences-bagdiff, 2
- occurrences-bagint, 2
- occurrences-cons, 2
- occurrences-delete, 2
- occurrences-delete2, 82
- occurrences-delete2-off, 94
- occurrences-in-a-set, 2
- open-plus, 5
- open-plus-off, 15
- or-zerop-tree, 45, 47, 49
- or-zerop-tree-is-not-zerop-tree, 47
 - off, 51
- permutation, 1, 3, 4, 83, 86–88
- permutation-a-b, 83
- permutation-a-b-off, 94
- permutation-append-arg1-arg2-bri
 - dge, 87
 - dge-off, 91
- permutation-append-split-by-parit
 - y-bridge, 88
 - y-bridge-off, 90
- permutation-as-subbagp, 83
- permutation-as-subbagp-helper, 83
- permutation-as-subbagp-helper-o
 - ff, 83
- permutation-as-subbagp-off, 83
- permutation-bagdiff, 87
- permutation-bagdiff-append, 87
- permutation-bagdiff-append-helpe
 - r, 87
 - r-off, 91
- permutation-bagdiff-append-off, 91
- permutation-bagdiff-off, 91
- permutation-delete-delete, 3
- permutation-does-not-effect-rpl
 - us, 86
 - us-off, 89
- permutation-implies-membership-eq
 - uivalence, 3
- permutation-reflexivity, 3
- permutation-right-cons, 3
- permutation-right-cons1, 3
- permutation-transitive, 86
- permutation-transitive-off, 92
- plus-add1, 5
- plus-cancellation, 6
- plus-cancellation-off, 15
- plus-difference-arg1, 6
- plus-difference-arg1-casesplit, 6
- plus-difference-arg2, 6
- plus-difference-arg2-casesplit, 6
- plus-difference-difference, 6
- plus-difference-difference-case
 - split, 6
- plus-fringe, 8–10, 12
- plus-remainder-times-quotient, 18
- plus-remainder-times-quotient-o
 - ff, 22
- plus-right-id2, 5
- plus-stopper, 5
- plus-tree, 8–10, 12
- plus-tree-bagdiff, 8
- plus-tree-bagdiff-off, 13
- plus-tree-delete, 8

- plus-tree-delete-off, 13
- plus-tree-plus-fringe, 9
- plus-tree-plus-fringe-off, 13
- quotient-1-arg1, 26
- quotient-1-arg1-casesplit, 26
- quotient-1-arg2, 25
- quotient-add1, 23
- quotient-difference-lessp-arg2, 39
- quotient-difference-lessp-arg2-off, 43
- quotient-difference1, 24
- quotient-difference2, 24
- quotient-difference3, 24
- quotient-exp, 25
- quotient-gcd-times-fact, 60
- quotient-gcd-times-fact-off, 76
- quotient-gcd-times-fact1, 60
- quotient-gcd-times-fact1-off, 76
- quotient-gcd-times-fact2, 60
- quotient-gcd-times-fact2-off, 76
- quotient-gcd-times-fact3, 60
- quotient-gcd-times-fact3-off, 76
- quotient-gcd-times-fact4, 60
- quotient-gcd-times-fact4-off, 76
- quotient-gcd-times-fact5, 61
- quotient-gcd-times-fact5-off, 76
- quotient-lessp-arg1, 24
- quotient-noop, 22
- quotient-of-non-number, 22
- quotient-plus, 23
- quotient-plus-fact, 25
- quotient-plus-proof, 23
- quotient-plus-times-times, 25
- quotient-plus-times-times-instance, 25
- quotient-plus-times-times-proof, 25
- quotient-quotient, 25
- quotient-remainder, 25
- quotient-remainder-instance, 25
- quotient-remainder-times, 24
- quotient-remainder-uniqueness, 38
- quotient-sub1, 23
- quotient-times, 23
- quotient-times-instance, 24
- quotient-times-instance-temp, 23
- quotient-times-instance-temp-of-f, 26
- quotient-times-instance-temp-proof, 23
- quotient-times-times, 24
- quotient-times-times-proof, 24
- quotient-x-x, 26
- quotient-zero, 23
- quotients, 26
- r1, 80
- r2, 96
- rational, 52–54, 62, 63, 81, 84, 86, 87
- rational-defns, 52
- rational-formp, 52, 63
- rational-generalization, 56
- rational-generalization-off, 77
- rational-ineg-numerator-reduce-bridge, 62
- rational-ineg-numerator-reduce-bridge-off, 74
- rationalp, 52–56, 61–64, 67
- rationalp-fix-rational, 55
- rationalp-fix-rational-off, 78
- rationalp-means, 63
- rationalp-means-off, 74
- rationalp-off, 80
- rationalp-rdifference, 55
- rationalp-rdifference-off, 78
- rationalp-reduce, 55
- rationalp-reduce-off, 78
- rationalp-rmagnitude, 55
- rationalp-rmagnitude-off, 78
- rationalp-rneg, 55
- rationalp-rneg-off, 78
- rationalp-rplus, 55
- rationalp-rplus-off, 78
- rationalp-rquotient, 55
- rationalp-rquotient-off, 78
- rationalp-rtimes, 55
- rationalp-rtimes-off, 78

- rationalp-simple-rplus, 64
- rationalp-simple-rplus-off, 73
- rationalp-simple-rtimes, 67
- rationalp-simple-rtimes-off, 70
- rdifference, 53, 55, 56, 65, 66, 68, 84–86
- rdifference-off, 80
- rdifference-rdifference-arg1, 66
- rdifference-rdifference-arg1-of f, 71
- rdifference-rdifference-arg2, 66
- rdifference-rdifference-arg2-of f, 71
- rdifference-reduce, 84
- rdifference-reduce-off, 93
- rdifference-rneg-arg2, 68
- rdifference-rneg-arg2-off, 69
- rdifference-rplus-hack, 84
- rdifference-rplus-hack-off, 93
- rdifference-rplus-hack2, 84
- rdifference-rplus-hack2-off, 93
- rdifference-x-x, 84
- rdifference-x-x-off, 93
- reduce, 53–56, 61–63, 65, 67, 81, 82, 84, 85
- reduce-0, 63
- reduce-0-off, 74
- reduce-difference, 65
- reduce-difference-off, 72
- reduce-eval\$rplus-tree, 82
- reduce-eval\$rplus-tree-off, 90
- reduce-nrationalp, 62
- reduce-nrationalp-off, 74
- reduce-off, 80
- reduce-reduce, 61
- reduce-reduce-off, 75
- reduce-rmagnitude, 65
- reduce-rmagnitude-off, 72
- reduce-rneg, 65
- reduce-rneg-off, 72
- reduce-rplus, 65
- reduce-rplus-off, 72
- reduce-rquotient, 65
- reduce-rquotient-off, 72
- reduce-rtimes, 65
- reduce-rtimes-off, 72
- remainder-0-sufficiency, 58
- remainder-0-sufficiency-off, 59
- remainder-0-sufficiency-off1, 76
- remainder-1-arg1, 21
- remainder-1-arg2, 21
- remainder-add1, 19
- remainder-difference1, 20
- remainder-difference2, 20
- remainder-difference3, 20
- remainder-difference3-off, 22
- remainder-equals-its-first-argument, 24
- ment-off, 26
- remainder-exp, 21
- remainder-exp-exp, 22
- remainder-gcd, 29
- remainder-noop, 18
- remainder-of-non-number, 18
- remainder-plus, 19
- remainder-plus-fact, 21
- remainder-plus-proof, 19
- remainder-plus-times-times, 21
- remainder-plus-times-times-instance, 21
- remainder-plus-times-times-proof, 21
- remainder-quotient-elim, 18
- remainder-remainder, 21
- remainder-sub1, 18
- remainder-times-times, 20
- remainder-times-times-off, 22
- remainder-times-times-proof, 19
- remainder-times1, 19
- remainder-times1-instance, 19
- remainder-times1-instance-proof, 19
- remainder-times1-proof, 19
- remainder-times2, 20
- remainder-times2-instance, 20
- remainder-times2-proof, 20
- remainder-x-x, 21
- remainder-zero, 18
- remainders, 22

- requal, 54, 61–68, 84, 85
- requal-associativity-of-rplus, 65
- requal-associativity-of-rplus-off, 72
- requal-associativity-of-rtimes, 67
- requal-associativity-of-rtimes-off, 70
- requal-associativity-of-simple-rplus, 64
- requal-associativity-of-simple-rplus-off, 73
- requal-off, 79
- requal-reduce-reduce-equal, 61
- requal-reduce-reduce-equal-off, 61
- requal-reduce-reduce-equal-off1, 75
- requal-reduce1, 61
- requal-reduce1-off, 75
- requal-reduce2, 61
- requal-reduce2-off, 75
- requal-rneg-rneg, 63
- requal-rneg-rneg-off, 74
- requal-rneg-simple-rplus, 66
- requal-rneg-simple-rplus-off, 71
- requal-rplus-rneg, 66
- requal-rplus-rneg-off, 71
- requal-rplus-x-simple-rneg-x, 84
- requal-rplus-x-simple-rneg-x-off, 93
- requal-rtimes-rneg, 68
- requal-rtimes-rneg-off, 69
- requal-rtimes-rplus-bridge, 68
- requal-rtimes-rplus-bridge-off, 69
- requal-rzerop-simple-rplus, 67
- requal-rzerop-simple-rplus-off, 69
- requal-simple-rplus-bridge, 65
- requal-simple-rplus-bridge-off, 72
- requal-simple-rplus-reduce-arg1, 62
- requal-simple-rplus-reduce-arg1-off, 75
- requal-simple-rplus-reduce-arg2, 62
- requal-simple-rplus-reduce-arg2-off, 75
- requal-simple-rplus-x-simple-rneg-x, 84
- requal-simple-rplus-x-simple-rneg-x-off, 93
- requal-simple-rplus-x-x-rewrite, 85
- requal-simple-rplus-x-x-rewrite-off, 92
- requal-simple-rtimes-bridge, 67
- requal-simple-rtimes-bridge-off, 70
- requal-simple-rtimes-reduce-arg1-off, 70
- requal-simple-rtimes-reduce-arg2-off, 70
- requal-simple-rtimes-reduce-arg1, 67
- requal-simple-rtimes-reduce-arg2, 67
- requal-simple-rtimes-requal-arg1-off, 70
- requal-simple-rtimes-requal-arg2-off, 70
- requal-simple-rtimes-requal-arg1, 66
- requal-simple-rtimes-requal-arg2, 67
- requal-simple-rtimes-simple-rneg-g, 68
- requal-simple-rtimes-simple-rneg-g-off, 69
- requal-simple-rtimes-simple-rpl-us-arg1, 68
- requal-simple-rtimes-simple-rpl-us-arg1-off, 69
- requal-simple-rtimes-simple-rpl-us-when-not-rzerop, 67
- requal-simple-rtimes-simple-rpl-us-when-not-rzerop-off, 70
- requal-x-x, 62
- requal-x-x-off, 75
- rinverse, 54
- rlessp, 54
- rlessp-off, 79
- rmagnitude, 54–56, 65
- rmagnitude-off, 79
- rneg, 53–56, 63, 65, 66, 68, 84
- rneg-off, 80
- rneg-rdifference, 68
- rneg-rdifference-off, 69
- rneg-reduce, 63
- rneg-reduce-off, 74
- rneg-rneg, 63
- rneg-rneg-off, 74
- rneg-rplus, 66
- rneg-rplus-off, 71
- rneqs-cancel-list, 87
- rneqs-cancel-list-off, 89
- rneqs-not-car-split-by-parity, 88
- rneqs-not-car-split-by-parity-off, 90
- rplus, 53, 55, 56, 61–63, 65, 66, 68, 81, 82, 84, 85

- rplus-eval\$rplus-tree, 82
- rplus-eval\$rplus-tree-off, 89
- rplus-fringe, 81, 88, 95
- rplus-off, 80
- rplus-open-up, 61
- rplus-open-up-off, 62
- rplus-open-up-off1, 75
- rplus-rdifference-arg1, 65
- rplus-rdifference-arg1-off, 72
- rplus-rdifference-arg2, 65
- rplus-rdifference-arg2-off, 72
- rplus-rdifference-hack, 84
- rplus-rdifference-hack-off, 93
- rplus-reduce-arg1, 62
- rplus-reduce-arg1-off, 75
- rplus-reduce-arg1-rewrite, 65
- rplus-reduce-arg1-rewrite-off, 72
- rplus-reduce-arg2, 62
- rplus-reduce-arg2-off, 75
- rplus-reduce-arg2-rewrite, 65
- rplus-reduce-arg2-rewrite-off, 71
- rplus-requal-arg1, 62
- rplus-requal-arg1-off, 62
- rplus-requal-arg1-off1, 75
- rplus-rneg-arg1, 66
- rplus-rneg-arg1-off, 71
- rplus-rneg-arg2, 66
- rplus-rneg-arg2-off, 71
- rplus-rzerop, 81
- rplus-rzerop-bridge, 81
- rplus-rzerop-bridge-off, 95
- rplus-rzerop-bridge2, 81
- rplus-rzerop-bridge2-off, 95
- rplus-rzerop-off, 95
- rplus-tree, 80–82, 86–88, 95
- rplus-x-rneg-x, 84
- rplus-x-rneg-x-off, 93
- rquotient, 54–56, 65
- rquotient-off, 79
- rtimes, 54–56, 65–68, 89
- rtimes-off, 79
- rtimes-rdifference-arg1, 68
- rtimes-rdifference-arg1-off, 69
- rtimes-rdifference-arg2, 68
- rtimes-rdifference-arg2-off, 69
- rtimes-rneg-arg1, 68
- rtimes-rneg-arg1-off, 69
- rtimes-rneg-arg2, 68
- rtimes-rneg-arg2-off, 69
- rtimes-rplus-arg1, 68
- rtimes-rplus-arg1-off, 69
- rtimes-rplus-arg2, 68
- rtimes-rplus-arg2-off, 69
- rzerop, 54, 66–68, 81
- rzerop-off, 71
- setp, 1, 2, 4
- setp-delete, 4
- setp-permutation, 4
- setp-permutation-base-case, 4
- setp-permutation-induction, 4
- setp-permutation-induction-step, 4
- sets-and-bags, 5
- simple-rinverse, 54
- simple-rmagnitude, 54
- simple-rmagnitude-off, 79
- simple-rneg, 53, 63, 66, 68, 84
- simple-rneg-off, 80
- simple-rneg-reduce, 63
- simple-rneg-reduce-off, 74
- simple-rneg-simple-rneg, 63
- simple-rneg-simple-rneg-off, 74
- simple-rplus, 53, 62–68, 84, 85
- simple-rplus-bridge, 68
- simple-rplus-bridge-off, 69
- simple-rplus-fix-rational-arg1, 63
- simple-rplus-fix-rational-arg1-off, 73
- simple-rplus-fix-rational-arg2, 63
- simple-rplus-fix-rational-arg2-off, 73
- simple-rplus-off, 80
- simple-rplus-rzerop, 67
- simple-rplus-rzerop-off, 70
- simple-rtimes, 53, 54, 66–68
- simple-rtimes-off, 80
- simple-rtimes-rzerop, 66
- simple-rtimes-rzerop-off, 71

- single-number-induction, 29
- split-by-parity, 81, 88, 89, 95
- stupid-hack, 46
- stupid-hack-off, 51
- subbagp, 1, 2, 4, 5, 8, 9, 36, 47–50, 82–84, 87, 88
- subbagp-append, 82
- subbagp-append-bridge, 88
- subbagp-append-bridge-off, 90
- subbagp-append-bridge2, 88
- subbagp-append-bridge2-off, 90
- subbagp-append-off, 94
- subbagp-append-simplify1, 88
- subbagp-append-simplify1-off, 91
- subbagp-append-simplify2, 88
- subbagp-append-simplify2-off, 90
- subbagp-bagdiff, 87
- subbagp-bagdiff-off, 91
- subbagp-bagint1, 5
- subbagp-bagint2, 5
- subbagp-cdr1, 5
- subbagp-cdr2, 5
- subbagp-delete, 4
- subbagp-delete-car, 83
- subbagp-delete-car-off, 94
- subbagp-delete-car2, 83
- subbagp-delete-car2-off, 94
- subbagp-delete-same, 83
- subbagp-delete-same-means, 83
- subbagp-delete-same-means-off, 94
- subbagp-delete-same-off, 94
- subbagp-implies-plus-tree-greater, 8
 - reqp-off, 13
- subbagp-make-negs, 84
- subbagp-make-negs-off, 93
- subbagp-necc, 83
- subbagp-necc-off, 94
- subbagp-permutation, 83
- subbagp-permutation-equiv, 88
- subbagp-permutation-equiv-off, 90
- subbagp-permutation-off, 94
- subbagp-transitive, 83
- subbagp-transitive-bridge, 87
- subbagp-transitive-bridge-helper, 87
 - r-off, 91
- subbagp-transitive-bridge-off, 91
- subbagp-transitive-off, 94
- subbagp-wit-lemma, 82
- subbagp-wit-lemma-off, 82
- subbagp-x-x, 88
- subbagp-x-x-off, 90
- times-1-arg1, 16
- times-add1, 15
- times-distributes-over-difference, 16
 - ce-proof, 16
- times-distributes-over-plus, 15
- times-distributes-over-plus-proof, 15
- times-fringe, 45, 47–50
- times-gcd-fact, 60
- times-gcd-fact-off, 76
- times-quotient, 16
- times-quotient-proof, 16
- times-sub1, 15
- times-tree, 45–50
- times-tree-append, 47
- times-tree-append-off, 51
- times-tree-of-times-fringe, 47
- times-tree-of-times-fringe-off, 51
- times-zero, 15
- transitivity-of-divides, 21
- transitivity-of-permutation, 4
- transitivity-of-permutation-base-case, 4
- transitivity-of-permutation-induction, 4
 - uction-step, 4
- transitivity-of-requal, 64
- transitivity-of-requal-bridge, 64
- transitivity-of-requal-bridge-off, 64
 - ff1, 73
- transitivity-of-requal-off, 73

zerop-makes-equal-true-bridge, 49
zerop-makes-equal-true-bridge-off, 52
zerop-makes-lessp-false-bridge, 48
zerop-makes-lessp-false-bridge-off, 52
zerop-makes-quotient-zero-bridge, 48
-off, 52
zerop-makes-times-tree-zero, 47
zerop-makes-times-tree-zero-off, 51
zerop-makes-times-tree-zero2, 47
zerop-makes-times-tree-zero2-off, 51