EVENT: Start with the library "interpreter".

;;; List Operations

THEOREM: car-append

\[
\text{car}(\text{append}(a, b)) = \begin{cases} 
\text{if listp}(a) \text{ then } \text{car}(a) \\
\text{else } \text{car}(b) \end{cases}
\]

THEOREM: listp-append

\[
\text{listp}(\text{append}(a, b)) = (\text{listp}(a) \lor \text{listp}(b))
\]

THEOREM: length-append

\[
\text{length}(\text{append}(a, b)) = (\text{length}(a) + \text{length}(b))
\]

DEFINITION:

\[
\text{plistp}(\text{list}) = \begin{cases} 
\text{if listp}(\text{list}) \text{ then } \text{plistp}(\text{cdr}(\text{list})) \\
\text{else } \text{list} = \text{nil} \end{cases}
\]

THEOREM: plistp-append-plistp

\[
\text{plistp}(\text{append}(a, b)) = \text{plistp}(b)
\]

THEOREM: append-plistp-nil

\[
(\text{append}(a, \text{nil}) = a) = \text{plistp}(a)
\]

THEOREM: not-lessp-count-append

\[
(\text{count}(x) + \text{count}(y)) \not< \text{count}(\text{append}(x, y))
\]

DEFINITION:

\[
\text{all-numberps}(\text{list}) = \begin{cases} 
\text{if listp}(\text{list}) \text{ then } \text{if } \text{car}(\text{list}) \in \text{N} \text{ then } \text{all-numberps}(\text{cdr}(\text{list})) \\
\text{else } \text{f} \end{cases} \text{endif} \\
\text{else } \text{t} \text{ endif}
\]

THEOREM: all-numberps-implies

\[
(\text{all-numberps}(\text{list}) \land (e \in list)) \rightarrow (e \in \text{N})
\]

;;; Set Operations

DEFINITION:

\[
\text{setp}(\text{list}) = \begin{cases} 
\text{if listp}(\text{list}) \text{ then } \text{if } \text{car}(\text{list}) \in \text{cdr}(\text{list}) \text{ then } \text{f} \\
\text{else setp}(\text{cdr}(\text{list})) \text{ endif} \\
\text{else } \text{t} \text{ endif}
\]

Theorem: setp-append

\[ (\neg \text{setp}(a)) \vee (\neg \text{setp}(b)) \rightarrow (\neg \text{setp}(\text{append}(a, b))) \]

Theorem: setp-member

\[ (x \in a) \land (x \in b) \rightarrow (\neg \text{setp}(\text{append}(a, b))) \]

Theorem: setp-append-cons

\[ \text{setp}(\text{append}(a, \text{cons}(x, b))) = \text{setp}(\text{cons}(x, \text{append}(a, b))) \]

Theorem: setp-append-not-listp

\[ (\neg \text{listp}(b)) \rightarrow (\text{setp}(\text{append}(a, b)) = \text{setp}(a)) \]

Theorem: setp-append-canonicalize

\[ \text{setp}(\text{append}(a, b)) = \text{setp}(\text{append}(b, a)) \]

Theorem: setp-member-1

\[ (\text{setp}(\text{append}(a, b)) \land (x \in b)) \rightarrow (x \not\in a) \]

Theorem: setp-member-2

\[ (\text{setp}(\text{append}(a, b)) \land (x \in a)) \rightarrow (x \not\in b) \]

;;; Subset Operations

Definition:

\[ \text{sublistp}(\text{sub}, \text{list}) = \text{if} \ \text{listp}(\text{sub}) \ \text{then} \ (\text{car}(\text{sub}) \in \text{list}) \land \text{sublistp}(\text{cdr}(\text{sub}), \text{list}) \ \text{else} \ \text{t endif} \]

Theorem: sublistp-append

\[ \text{sublistp}(\text{append}(a, b), \text{list}) = (\text{sublistp}(a, \text{list}) \land \text{sublistp}(b, \text{list})) \]

Theorem: member-of-sublistp-is-member

\[ ((a \in b) \land \text{sublistp}(b, c)) \rightarrow (a \in c) \]

Theorem: sublistp-of-sublistp-is-sublistp

\[ (\text{sublistp}(a, b) \land \text{sublistp}(b, c)) \rightarrow \text{sublistp}(a, c) \]

Theorem: sublistp-normalize

\[ (\neg \text{plistp}(b)) \rightarrow (\text{sublistp}(a, b) = \text{sublistp}(a, \text{append}(b, \text{nil}))) \]

Definition:

\[ \text{sei}(a, b) = \text{if} \ \text{listp}(a) \ \text{then} \ \text{sei}(\text{cdr}(a), \text{append}(b, \text{list}(\text{car}(a)))) \ \text{else} \ \text{t endif} \]
THEOREM: sublistp-easy
sublistp (a, append (b, a))

THEOREM: sublistp-reflexive
sublistp (a, a)

THEOREM: sublistp-in-append
(sublistp (x, a) ∨ sublistp (x, b)) → sublistp (x, append (a, b))

THEOREM: sublistp-in-cons
sublistp (a, y) → sublistp (a, cons (x, y))

;; Tree Operations

DEFINITION:
nodes-rec (flag, tree) =
  if listp (tree)
    then if flag = ’tree
      then cons (car (tree), nodes-rec (’forest, cdr (tree)))
      else append (nodes-rec (’tree, car (tree)),
                  nodes-rec (’forest, cdr (tree)))
    else nil endif
  else nil endif

DEFINITION: nodes (tree) = nodes-rec (’tree, tree)

DEFINITION:
roots (forest) =
  if listp (forest) then cons (caar (forest), roots (cdr (forest)))
  else forest endif

DEFINITION:
children-rec (flag, node, tree) =
  if listp (tree)
    then if flag = ’tree
      then if car (tree) = node
          then append (roots (cdr (tree)),
                      children-rec (’forest, node, cdr (tree)))
          else children-rec (’forest, node, cdr (tree)) endif
        else children-rec (’tree, node, car (tree)),
        children-rec (’forest, node, cdr (tree))) endif
    else nil endif

DEFINITION:
children (node, tree) = children-rec (’tree, node, tree)
DEFINITION:
pARENT-REC (flag, node, tree) =
  if listp (tree)
    then if flag = 'tree
      then if node ∈ roots (cdr (tree))
        then cons (car (tree), parent-rec ('forest, node, cdr (tree)))
        else parent-rec ('forest, node, cdr (tree)) endif
      else parent-rec ('tree, node, car (tree)),
        parent-rec ('forest, node, cdr (tree)) endif
    else nil endif
  else append (parent-rec ('tree, node, car (tree)),
    parent-rec ('forest, node, cdr (tree))) endif

DEFINITION:
parent (node, tree) = car (parent-rec ('tree, node, tree))

DEFINITION:
proper-tree (flag, tree) =
  if flag = 'tree
    then if listp (tree) then proper-tree ('forest, cdr (tree))
    else f endif
  elseif listp (tree)
    then proper-tree ('tree, car (tree)) ∧ proper-tree ('forest, cdr (tree))
    else tree = nil endif

THEOREM: canonicalize-nodes-rec-flag
nodes-rec (flag, tree) =
  if flag = 'tree then nodes-rec ('tree, tree)
  else nodes-rec ('forest, tree) endif

THEOREM: canonicalize-proper-tree-flag
proper-tree (flag, tree) =
  if flag = 'tree then proper-tree ('tree, tree)
  else proper-tree ('forest, tree) endif

THEOREM: canonicalize-parent-rec-flag
parent-rec (flag, child, tree) =
  if flag = 'tree then parent-rec ('tree, child, tree)
  else parent-rec ('forest, child, tree) endif

THEOREM: canonicalize-children-rec-flag
children-rec (flag, parent, tree) =
  if flag = 'tree then children-rec ('tree, parent, tree)
  else children-rec ('forest, parent, tree) endif

THEOREM: not-flag-tree
\[(\text{flag} \neq \text{tree}) \land (\text{flag} \neq \text{forest})\]
\[\rightarrow (\text{nodes-rec}(\text{flag}, \text{tree}) = \text{nodes-rec}(\text{forest}, \text{tree}))\]
\[\land (\text{proper-tree}(\text{flag}, \text{tree}) = \text{proper-tree}(\text{forest}, \text{tree}))\]
\[\land (\text{parent-rec}(\text{flag}, \text{child}, \text{tree}) = \text{parent-rec}(\text{forest}, \text{child}, \text{tree}))\]
\[\land (\text{children-rec}(\text{flag}, \text{parent}, \text{tree}) = \text{children-rec}(\text{forest}, \text{parent}, \text{tree}))\]

**Theorem:** parent-rec-children-rec

\[(\text{child} \in \text{children-rec}(\text{flag}, \text{parent}, \text{tree})) = (\text{parent} \in \text{parent-rec}(\text{flag}, \text{child}, \text{tree}))\]

**Event:** Disable parent-rec-children-rec.

**Theorem:** plistp-children-rec

\[\text{plistp}(\text{children-rec}(\text{flag}, \text{parent}, \text{tree}))\]

**Theorem:** plistp-parent-rec

\[\text{plistp}(\text{parent-rec}(\text{flag}, \text{child}, \text{tree}))\]

**Theorem:** plistp-roots

\[\text{proper-tree}(\text{forest}, \text{forest}) \rightarrow \text{plistp}(\text{roots}(\text{forest}))\]

**Theorem:** member-roots-member-forest

\[(\text{proper-tree}(\text{forest}, \text{forest}) \land (\text{node} \in \text{roots}(\text{forest}))) \rightarrow (\text{node} \in \text{nodes-rec}(\text{forest}, \text{forest}))\]

**Theorem:** not-member-no-parent

\[(\text{proper-tree}(\text{flag}, \text{tree}) \land (\text{node} \notin \text{nodes-rec}(\text{flag}, \text{tree}))) \rightarrow (\text{parent-rec}(\text{flag}, \text{node}, \text{tree}) = \text{nil})\]

**Theorem:** member-child-tree

\[(\text{proper-tree}(\text{flag}, \text{tree}) \land (\text{child} \in \text{children-rec}(\text{flag}, \text{node}, \text{tree}))) \rightarrow (\text{child} \in \text{nodes-rec}(\text{flag}, \text{tree}))\]

**Theorem:** setp-tree-unique-parent

\[(\text{proper-tree}(\text{flag}, \text{tree}) \land \text{setp}(\text{nodes-rec}(\text{flag}, \text{tree}))) \rightarrow (\text{parent-rec}(\text{flag}, \text{child}, \text{tree}) = \left\{\begin{array}{ll}
\text{if} \ \text{child} \in \text{nodes-rec}(\text{flag}, \text{tree}) \text{ then if } ((\text{flag} = \text{tree}) \land (\text{car}(\text{tree}) = \text{child})) \lor ((\text{flag} \neq \text{tree}) \land \text{child} \in \text{roots}(\text{tree})) \text{ then nil else list(\text{car}(\text{parent-rec}(\text{flag}, \text{child}, \text{tree}))) endif else nil endif}
\end{array}\right.)\]

**Event:** Disable setp-tree-unique-parent.
Theorem: member-parent-parent
(proper-tree (flag, tree)
∧ setp (nodes-rec (flag, tree))
∧ (parent ∈ parent-rec (flag, child, tree)))
→ (parent-rec (flag, child, tree) = list (parent))

Theorem: parent-of-child
(proper-tree (flag, tree)
∧ setp (nodes-rec (flag, tree))
∧ (child ∈ children-rec (flag, parent, tree)))
→ (parent-rec (flag, child, tree) = list (parent))

Theorem: member-parent-member-tree
(parent ∈ parent-rec (flag, child, tree))
→ (parent ∈ nodes-rec (flag, tree))

Theorem: node-that-has-child-is-in-tree
listp (children-rec (flag, parent, tree)) → (parent ∈ nodes-rec (flag, tree))

Theorem: node-that-has-parent-is-in-tree
(proper-tree (flag, tree) ∧ listp (parent-rec (flag, child, tree)))
→ (child ∈ nodes-rec (flag, tree))

Theorem: sublistp-children-generalized
(proper-tree (flag, tree)
∧ sublistp (children, children-rec (flag, parent, tree)))
→ sublistp (children, nodes-rec (flag, tree))


Theorem: sublistp-children
proper-tree (flag, tree)
→ sublistp (children-rec (flag, parent, tree), nodes-rec (flag, tree))

Definition:
subtreep (flag, subtree, tree)
= if listp (tree) ∧ listp (subtree)
   then if flag = 'tree
       then if subtree = tree then t
         else subtreep ('forest, subtree, cdr (tree)) endif
     elseif subtreep ('tree, subtree, car (tree)) then t
         else subtreep ('forest, subtree, cdr (tree)) endif
   else f endif
Definition:
subtrees (flag, tree) =
  if listp (tree)
    then if flag = 'tree
      then cons (tree, subtrees ('forest, cdr (tree)))
      else append (subtrees ('tree, car (tree)),
                   subtrees ('forest, cdr (tree)))
    else nil
  endif
else nil endif

Theorem: subtreep-subtrees
(subtree ∈ subtrees (flag, tree)) → subtreep (flag, subtree, tree)

Definition:
next-level (subtrees) =
  if listp (subtrees)
    then append (cdar (subtrees), next-level (cdr (subtrees)))
    else subtrees
  endif

Theorem: nodes-rec-forest-append
nodes-rec ('forest, append (a, b)) =
  append (nodes-rec ('forest, a), nodes-rec ('forest, b))

Theorem: next-level-reduces-count
listp (subtrees) → (count (next-level (subtrees)) < count (subtrees))

Theorem: next-level-of-tree-in-subtrees
proper-tree ('forest, forest) → sublistp (forest, subtrees ('forest, forest))

Theorem: subtrees-of-subtree-in-complete-subtrees
(proper-tree ('tree, subtree) ∧ (subtree ∈ subtrees (flag, tree)))
→ sublistp (subtrees ('tree, subtree), subtrees (flag, tree))

Theorem: subtrees-of-subtrees-in-complete-subtrees
(proper-tree ('forest, subtrees) ∧ sublistp (subtrees, subtrees (flag, tree)))
→ sublistp (subtrees ('forest, subtrees), subtrees (flag, tree))

Theorem: next-level-in-subtrees-forest
proper-tree ('forest, subtrees)
→ sublistp (next-level (subtrees), subtrees ('forest, subtrees))

Theorem: next-level-of-subtrees-in-complete-subtrees
(proper-tree ('forest, subtrees) ∧ sublistp (subtrees, subtrees (flag, tree)))
→ sublistp (next-level (subtrees), subtrees (flag, tree))
Theorem: proper-tree-of-append
(proper-tree (’forest, a) ∧ proper-tree (’forest, b))
→ proper-tree (’forest, append (a, b))

Theorem: proper-tree-next-level-of-proper-tree
proper-tree (’forest, subtrees)
→ proper-tree (’forest, next-level (subtrees))

Theorem: not-member-subtrees
(root \not\in nodes-rec (flag, tree))
→ (cons (root, forest) \not\in subtrees (flag, tree))

Theorem: not-member-no-children
(parent \not\in nodes-rec (flag, tree))
→ (children-rec (flag, parent, tree) = nil)

Theorem: no-children-in-rest-of-forest
(setp (append (nodes-rec (’tree, tree), nodes-rec (’forest, forest)))
∧ (parent \in nodes-rec (’tree, tree)))
→ (children-rec (’forest, parent, forest) = nil)

Theorem: no-children-in-rest-of-tree
(setp (append (nodes-rec (’tree, tree), nodes-rec (’forest, forest)))
∧ (parent \in nodes-rec (’forest, forest)))
→ (children-rec (’tree, parent, tree) = nil)

Theorem: member-subtree-member-tree
(cons (root, forest) \in subtrees (flag, tree))
→ (root \in nodes-rec (flag, tree))

Theorem: children-of-setp-tree
(setp (nodes-rec (flag, tree))
∧ proper-tree (flag, tree)
∧ (cons (root, forest) \in subtrees (flag, tree)))
→ (children-rec (flag, root, tree) = roots (forest))

Theorem: node-has-parent
((node \in nodes-rec (flag, tree))
∧ proper-tree (flag, tree)
∧ if flag = ’tree then node \neq car (tree)
else node \not\in roots (tree) endif)
→ (car (parent-rec (flag, node, tree)) \in nodes-rec (flag, tree))

Theorem: parent-is-not-itself-generalized
(setp (nodes-rec (flag, tree))
∧ proper-tree (flag, tree)
∧ listp (parent-rec (flag, child, tree)))
→ (child \neq car (parent-rec (flag, child, tree)))
\section*{Theorem: parent-is-not-itself}
\begin{align*}
\text{setp} & (\text{nodes-rec} (\text{'tree}, \text{tree})) \\
\land & \text{proper-tree} (\text{'tree}, \text{tree}) \\
\land & (\text{child} \in \text{cdr} (\text{nodes-rec} (\text{'tree}, \text{tree}))) \\
\rightarrow & (\text{child} \neq \text{car} (\text{parent-rec} (\text{'tree}, \text{child}, \text{tree})))
\end{align*}

\section*{Theorem: listp-parent-rec-equals}
\begin{align*}
\text{setp} & (\text{nodes-rec} (\text{flag}, \text{tree})) \land \text{proper-tree} (\text{flag}, \text{tree}) \\
\rightarrow & \text{listp} (\text{parent-rec} (\text{flag}, \text{child}, \text{tree})) \\
& = ((\text{child} \in \text{nodes-rec} (\text{flag}, \text{tree})) \\
& \land \begin{cases} 
\text{if flag} = \text{'tree} \text{ then child} \neq \text{car} (\text{tree}) \\
\text{else child} \notin \text{roots} (\text{tree}) \end{cases})
\end{align*}

\section*{Theorem: parent-is-not-child}
\begin{align*}
\text{setp} & (\text{nodes-rec} (\text{flag}, \text{tree})) \\
\land & \text{proper-tree} (\text{flag}, \text{tree}) \\
\land & \text{listp} (\text{parent-rec} (\text{flag}, \text{child}, \text{tree})) \\
\rightarrow & (\text{car} (\text{parent-rec} (\text{flag}, \text{child}, \text{tree})) \notin \text{children-rec} (\text{flag}, \text{child}, \text{tree}))
\end{align*}

\section*{Theorem: parent-not-in-children}
\begin{align*}
\text{setp} & (\text{nodes-rec} (\text{'tree}, \text{tree})) \\
\land & \text{proper-tree} (\text{'tree}, \text{tree}) \\
\land & (\text{parent} \in \text{cdr} (\text{nodes-rec} (\text{'tree}, \text{tree}))) \\
\rightarrow & (\text{parent} \notin \text{children-rec} (\text{'tree}, \text{parent}, \text{tree}))
\end{align*}

\section*{Variables and channel operations}
\begin{align*}
\text{Definition:} & \quad \text{value} (\text{key}, \text{state}) = \text{cdr} \ (\text{assoc} (\text{key}, \text{state})) \\
\text{Definition:} & \quad \text{channel} (\text{name}, \text{state}) = \text{value} (\text{name}, \text{state}) \\
\text{Definition:} & \quad \text{empty} (\text{name}, \text{state}) = (\neg \text{listp} (\text{channel} (\text{name}, \text{state}))) \\
\text{Definition:} & \quad \text{head} (\text{name}, \text{state}) = \text{car} (\text{channel} (\text{name}, \text{state})) \\
\text{Definition:} & \quad \text{send} (\text{channel}, \text{message}, \text{state}) \\
& = \text{append} (\text{channel} (\text{channel}, \text{state}), \text{list} (\text{message})) \\
\text{Definition:} & \quad \text{receive} (\text{channel}, \text{state}) = \text{cdr} (\text{channel} (\text{channel}, \text{state}))
\end{align*}

\section*{Program Specific}
Definition:
\[ \text{status} (\text{node}, \text{state}) = \text{value} (\text{cons} (\text{status, node}), \text{state}) \]

Definition:
\[ \text{found-value} (\text{node}, \text{state}) = \text{value} (\text{cons} (\text{found-value, node}), \text{state}) \]

Definition:
\[ \text{outstanding} (\text{node}, \text{state}) = \text{value} (\text{cons} (\text{outstanding, node}), \text{state}) \]

Definition:
\[ \text{node-value} (\text{node}, \text{state}) = \text{value} (\text{cons} (\text{node-value, node}), \text{state}) \]

Definition:
\[
\begin{align*}
\text{send-find} & (\text{to-children}, \text{old}, \text{new}) \\
= & \quad \text{if listp} (\text{to-children}) \\
& \quad \text{then} (\text{channel} (\text{car} (\text{to-children}), \text{new}) \\
& \quad \quad = \quad \text{send} (\text{car} (\text{to-children}), \text{find}, \text{old}) \\
& \quad \quad \quad \text{∧ send-find} (\text{cdr} (\text{to-children}), \text{old}, \text{new}) \\
& \quad \quad \text{endif} \\
& \quad \text{else t endif}
\end{align*}
\]

;;; The four program statements

Definition:
\[
\begin{align*}
\text{receive-find} & (\text{old}, \text{new}, \text{node}, \text{from-parent}, \text{to-parent}, \text{to-children}) \\
= & \quad \text{if head} (\text{from-parent}, \text{old}) = \text{find} \\
& \quad \text{then} (\text{channel} (\text{from-parent}, \text{new}) = \text{receive} (\text{from-parent}, \text{old})) \\
& \quad \quad \text{∧ (status} (\text{node}, \text{new}) = \text{started}) \\
& \quad \quad \text{∧ (found-value} (\text{node}, \text{new}) = \text{node-value} (\text{node}, \text{old}) \text{)} \\
& \quad \quad \text{∧ (outstanding} (\text{node}, \text{new}) = \text{length} (\text{to-children}) \text{)} \\
& \quad \quad \text{∧ send-find} (\text{to-children}, \text{old}, \text{new}) \\
& \quad \quad \text{∧ (channel} (\text{to-parent}, \text{new}) \\
& \quad \quad \quad \text{= \quad if length} (\text{to-children}) \simeq 0 \\
& \quad \quad \quad \quad \text{then send} (\text{to-parent}, \text{node-value} (\text{node}, \text{old}), \text{old}) \\
& \quad \quad \quad \quad \text{else channel} (\text{to-parent}, \text{old}) \text{ endif} \\
& \quad \quad \text{∧ changed} (\text{old}, \\
& \quad \quad \quad \text{new}, \\
& \quad \quad \quad \text{append} (\text{list} (\text{from-parent}, \\
& \quad \quad \quad \text{to-parent}, \\
& \quad \quad \quad \text{cons} (\text{status}, \text{node}), \\
& \quad \quad \quad \text{cons} (\text{found-value}, \text{node}), \\
& \quad \quad \quad \text{cons} (\text{outstanding}, \text{node}), \\
& \quad \quad \quad \text{to-children}))) \\
& \quad \quad \text{else changed} (\text{old}, \text{new}, \text{nil}) \text{ endif}
\end{align*}
\]
Definition:
\[ \min(x, y) = \begin{cases} 
\text{fix}(x) & \text{if } x < y \\
\text{fix}(y) & \text{else} 
\end{cases} \]

Definition:
\[
\text{receive-report}(\text{old, new, node, from-child, to-parent}) = \\
\begin{cases} 
\text{changed}(\text{old, new, nil}) & \text{if empty(from-child, old)} \\
\text{channel(from-child, new) = receive(from-child, old)} & \text{else} 
\end{cases} \\
\quad \land (\text{found-value(node, new)} = \min(\text{found-value(node, old)}, \text{head(from-child, old)})) \\
\quad \land (\text{outstanding(node, new)} = (\text{outstanding(node, old)} - 1)) \\
\quad \land (\text{channel(to-parent, new)} = \\
\quad \quad \begin{cases} 
\text{send(to-parent),} \\
\text{found-value(node, new),} \\
\text{old} \end{cases} & \text{if outstanding(node, new) } \not\approx 0 \\
\quad \text{else channel(to-parent, old)} \end{cases} \]
\quad \land \text{changed(old, new, list(from-child, to-parent, cons('outstanding, node), cons('found-value, node)))} \]

Definition:
\[
\text{start}(\text{old, new, root, to-children}) = \\
\begin{cases} 
\text{changed(old, new, list(from-child, to-parent, cons('status, root), cons('found-value, root), cons('outstanding, root)), to-children))} & \text{if status(root, old) } \not\approx \text{'not-started} \\
\text{else changed(old, new, nil)} \end{cases} \]

Definition:
\[
\text{root-receive-report}(\text{old, new, root, from-child}) = \\
\begin{cases} 
\text{changed(old, new, nil)} & \text{if empty(from-child, old)} \\
\text{else} \end{cases} \\
\quad \land \text{changed(old, new, append(list(cons('status, root), cons('found-value, root), cons('outstanding, root)), to-children)))} \]

11
else (channel (from-child, new) = receive (from-child, old))
   ∧ (found-value (root, new)
       = min (found-value (root, old),
              head (from-child, old)))
   ∧ (outstanding (root, new)
       = (outstanding (root, old) - 1))
   ∧ changed (old,
              new,
              list (from-child,
                  cons ('outstanding, root),
                  cons ('found-value, root)))
endif

;;; The Program

**Definition:**
rfp (node, children)
= if listp (children)
   then cons (cons (node, car (children)), rfp (node, cdr (children)))
   else nil endif

**Definition:**
receive-find-prg (nodes, tree)
= if listp (nodes)
   then cons (list ('receive-find,
                   car (nodes),
                   cons (parent (car (nodes), tree), car (nodes)),
                   cons (car (nodes), parent (car (nodes), tree)),
                   rfp (car (nodes), children (car (nodes), tree)))))
       receive-find-prg (cdr (nodes), tree))
   else nil endif

**Theorem:** member-receive-find-prg
(statement ∈ receive-find-prg (nodes, tree))
= (12 (car (statement) = 'receive-find)
   ∧ (cadr (statement) ∈ nodes)
   ∧ listp (caddr (statement))
   ∧ (caaddr (statement) = parent (cadr (statement), tree))
   ∧ (cdaddr (statement) = cadr (statement))
   ∧ listp (cadaddr (statement))
   ∧ (caaddr (statement) = cadr (statement))
   ∧ (cdaddr (statement) = parent (cadr (statement), tree))
   ∧ (caddddr (statement) = rfp (cadr (statement), children (cadr (statement), tree)))
   ∧ (cadddddr (statement) = nil))
**Definition:**

\[ \text{rrp}(\text{node}, \text{children}, \text{parent}) = \begin{cases} \text{if } \text{listp}(\text{children}) \text{ then cons(list('receive-report,} \\
\text{node,} \\
\text{cons(car(\text{children}), \text{node}),} \\
\text{cons(\text{node, parent)}},) \\
\text{rrp(\text{node, cdr(\text{children}, \text{parent})})} \\
\text{else nil endif} \end{cases} \]

**Theorem:** member-rrp

\[ (\text{statement} \in \text{rrp}(\text{node}, \text{children}, \text{parent})) = ((\text{car(\text{statement})} = \text{'receive-report}) \wedge \text{listp(cadr(\text{statement}))} \wedge \text{listp(caddr(\text{statement}))} \wedge \text{(caaddr(\text{statement})} \in \text{children} \wedge \text{listp(cadddr(\text{statement}))} \wedge \text{caadddr(\text{statement})} = \text{nil}) \]

**Definition:**

\[ \text{receive-report-prg}(\text{nodes, tree}) = \begin{cases} \text{if } \text{listp}(\text{nodes}) \text{ then append(rrp(car(\text{nodes}),} \\
\text{children(car(\text{nodes), tree}),} \\
\text{parent(car(\text{nodes), tree)},) \\
\text{receive-report-prg(cdr(\text{nodes}, \text{tree}))} \\
\text{else nil endif} \end{cases} \]

**Theorem:** member-receive-report-prg

\[ (\text{statement} \in \text{receive-report-prg}(\text{nodes, tree})) = ((\text{car(\text{statement})} = \text{'receive-report}) \wedge \text{(caaddr(\text{statement})} \in \text{children(cadr(\text{statement}), tree))} \wedge \text{listp(cadddr(\text{statement}))} \wedge \text{caadddr(\text{statement})} = \text{cadr(\text{statement})}) \wedge \text{caadddr(\text{statement})} = \text{parent(cadr(\text{statement}), tree))} \wedge \text{cddddr(\text{statement})} = \text{nil}) \]

**Definition:**

\[ \text{start-prg}(\text{root, tree}) = \text{list(list('start, root, rfp(root, children(root, tree))))} \]
THEOREM: member-start-prg
\( (\text{statement} \in \text{start-prg}(\text{root}, \text{tree})) \)
\[= ((\text{car} (\text{statement}) = \text{'start}) \]
\(\land \ (\text{cadr} (\text{statement}) = \text{root}) \]
\(\land \ (\text{caddr} (\text{statement}) = \text{rfp} (\text{root}, \text{children} (\text{root}, \text{tree}))) \]
\(\land \ (\text{cdddr} (\text{statement}) = \text{nil})\)

DEFINITION:
\(\text{rrrp} (\text{root}, \text{children}) \)
\[= \begin{cases} \text{if} \ \text{listp} (\text{children}) \\
\quad \text{then} \ \text{cons} (\text{list} (\text{'root-receive-report}, \\
\quad \quad \text{root,} \\
\quad \quad \text{cons} (\text{car} (\text{children}), \text{root})), \\
\quad \text{rrrp} (\text{root}, \text{cdr} (\text{children}))) \\
\quad \text{else} \ \text{nil} \ \text{endif} \end{cases} \]

THEOREM: member-rrrp
\( (\text{statement} \in \text{rrrp}(\text{root}, \text{children})) \)
\[= ((\text{car} (\text{statement}) = \text{'root-receive-report}) \]
\(\land \ (\text{cadr} (\text{statement}) = \text{root}) \]
\(\land \ \text{listp} (\text{caddr} (\text{statement})) \]
\(\land \ (\text{caaddr} (\text{statement}) \in \text{children}) \]
\(\land \ (\text{cdaddr} (\text{statement}) = \text{root}) \]
\(\land \ (\text{cdddr} (\text{statement}) = \text{nil})\)

DEFINITION:
\(\text{root-receive-report-prg} (\text{root}, \text{tree}) = \text{rrrp} (\text{root}, \text{children} (\text{root}, \text{tree})) \)

THEOREM: member-root-receive-report-prg
\( (\text{statement} \in \text{root-receive-report-prg}(\text{root}, \text{tree})) \)
\[= ((\text{car} (\text{statement}) = \text{'root-receive-report}) \]
\(\land \ (\text{cadr} (\text{statement}) = \text{root}) \]
\(\land \ \text{listp} (\text{caddr} (\text{statement})) \]
\(\land \ (\text{caaddr} (\text{statement}) \in \text{children} (\text{root}, \text{tree})) \]
\(\land \ (\text{cdaddr} (\text{statement}) = \text{root}) \]
\(\land \ (\text{cdddr} (\text{statement}) = \text{nil})\)

DEFINITION:
\(\text{tree-prg} (\text{tree}) \)
\[= \text{append} (\text{start-prg} (\text{car} (\text{tree}), \text{tree}), \]
\(\text{append} (\text{root-receive-report-prg} (\text{car} (\text{tree}), \text{tree}), \]
\(\text{append} (\text{receive-find-prg} (\text{cdr} (\text{nodes} (\text{tree})), \text{tree}), \]
\(\text{receive-report-prg} (\text{cdr} (\text{nodes} (\text{tree})), \text{tree})))\))

THEOREM: equal-if
(if test then p1
else p2 endif)
= if test then r1
else r2 endif
= if test then p1 = r1
else p2 = r2 endif

THEOREM: member-tree-prg

(statement ∈ tree-prg(tree))
= (((car (statement) = 'start)
∧ (cadr (statement) = car (tree))
∧ (caddr (statement)
= rfp (car (tree), children (car (tree), tree)))
∧ (cdddr (statement) = nil))
∨ ((car (statement) = 'root-receive-report)
∧ (cadr (statement) = car (tree))
∧ listp (caddr (statement))
∧ (caaddr (statement) ∈ children (car (tree), tree))
∧ (cdaddr (statement) = car (tree))
∧ (cdddr (statement) = nil))
∨ ((car (statement) = 'receive-find)
∧ (cadr (statement) ∈ cdr (nodes (tree)))
∧ listp (caddr (statement))
∧ (caaddr (statement) = parent (cadr (statement), tree))
∧ (cdaddr (statement) = cadr (statement))
∧ listp (cadddr (statement))
∧ (caadddr (statement) = cadr (statement))
∧ (cdadddr (statement) = parent (cadr (statement), tree))
∧ (caddddr (statement) = rfp (cadr (statement),
children (cadr (statement), tree)))
∧ (cdddddr (statement) = nil))
∨ ((car (statement) = 'receive-report)
∧ (cadr (statement) ∈ cdr (nodes (tree)))
∧ listp (caddr (statement))
∧ (caaddr (statement) ∈ children (cadr (statement), tree))
∧ (cdaddr (statement) = cadr (statement))
∧ listp (cadddr (statement))
∧ (caadddr (statement) = cadr (statement))
∧ (cdadddr (statement) = parent (cadr (statement), tree))
∧ (caddddr (statement) = nil)))

;;; Correctness
Definition:
	
treep (tree) = (setp (nodes (tree))
\[\land\] all-numberps (nodes (tree))
\[\land\] proper-tree ('tree, tree))

Definition:
		total-outstanding (nodes, tree, state) =
	\begin{align*}
\text{if} & \quad \text{listp (nodes)} \\
\text{then} & \quad \text{total-outstanding (cdr (nodes), tree, state)} \\
\quad & \quad + \quad \text{if status (car (nodes), state) = 'started} \\
\quad & \quad \quad \quad \text{then} \quad \text{outstanding (car (nodes), state))} \\
\quad & \quad \quad \text{else} \quad 1 + \text{length (children (car (nodes), tree))} \\
\text{endif}
\text{else} & \quad 0
\end{align*}
endif

Definition:

dl (down-links, state) =
\begin{align*}
\text{if} & \quad \text{listp (down-links)} \\
\text{then} & \quad ((\text{empty (car (down-links), state)} \\
\quad \land \quad (\text{status (caar (down-links), state)}
\quad = \quad \text{status (cdar (down-links), state)))) \\
\quad \lor \quad ((\text{channel (car (down-links), state)} = \text{list ('find)}) \\
\quad \land \quad (\text{status (caar (down-links), state)} = 'started') \\
\quad \land \quad (\text{status (cdar (down-links), state)}
\quad = \quad 'not-started)))) \\
\quad \land \quad \text{dl (cdr (down-links), state)}
\text{else} & \quad t \text{ endif}
\end{align*}

Definition:

done (node, state) =
\begin{align*}
(\text{status (node, state)} = 'started) \\
\land \quad (\text{outstanding (node, state)} \equiv 0))
\end{align*}

Definition:
 ul (up-links, state) =
\begin{align*}
\text{if} & \quad \text{listp (up-links)} \\
\text{then} & \quad (\text{empty (car (up-links), state)} \\
\quad \lor \quad ((\text{channel (car (up-links), state)}
\quad = \quad \text{list (found-value (caar (up-links), state)))} \\
\quad \land \quad \text{done (caar (up-links), state))} \\
\quad \land \quad \text{ul (cdr (up-links), state)}
\text{else} & \quad t \text{ endif}
\end{align*}

Definition:
reporte (node, parent, state) =
\begin{align*}
(\text{done (node, state)} \land \text{empty (cons (node, parent), state))}
\end{align*}
Definition:
number-not-reported (children, parent, state)
= if listp (children)
  then if reported (car (children), parent, state)
    then number-not-reported (cdr (children), parent, state)
    else 1 + number-not-reported (cdr (children), parent, state) endif
  else 0 endif

Definition:
min-of-reported (children, parent, state, min)
= if listp (children)
  then if reported (car (children), parent, state)
    then min (found-value (car (children), state),
               min-of-reported (cdr (children), parent, state, min))
    else min-of-reported (cdr (children), parent, state, min) endif
  else min endif

Definition:
no (nodes, tree, state)
= if listp (nodes)
  then if status (car (nodes), state) = 'started
    then (outstanding (car (nodes), state) =
          number-not-reported (children (car (nodes), tree),
                               car (nodes),
                               state))
          ∧ (found-value (car (nodes), state) =
              min-of-reported (children (car (nodes), tree),
                                 car (nodes),
                                 state,
                                 node-value (car (nodes), state)))
    else t endif
    ∧ no (cdr (nodes), tree, state)
  else t endif

Definition:
down-links-1 (parent, children)
= if listp (children)
  then cons (cons (parent, car (children)),
            down-links-1 (parent, cdr (children)))
  else nil endif

Definition:
down-links (nodes, tree)
= if listp (nodes)
  then append (down-links-1 (car (nodes), children (car (nodes), tree)),
               down-links-1 (cdr (nodes), children (cdr (nodes), tree)))
  else nil endif
down-links (cdr (nodes, tree))
else nil endif

DEFINITION:
up-links (nodes, tree)
  = (if listp (nodes)
      then cons (cons (car (nodes), parent (car (nodes), tree)),
                  up-links (cdr (nodes), tree))
      else nil endif

DEFINITION:
inv (tree, state)
  = (dl (down-links (nodes (tree), tree), state)
       ∧ ul (up-links (cdr (nodes (tree)), tree), state)
       ∧ no (nodes (tree), tree, state))

DEFINITION:
not-started (nodes, state)
  = (if listp (nodes)
      then (status (car (nodes), state) = 'not-started)
           ∧ not-started (cdr (nodes), state)
      else t endif

DEFINITION:
all-channels (tree)
  = append (up-links (cdr (nodes (tree)), tree), down-links (nodes (tree), tree))

DEFINITION:
all-empty (channels, state)
  = (if listp (channels)
      then empty (car (channels), state) ∧ all-empty (cdr (channels), state)
      else t endif

DEFINITION:
min-node-value (nodes, state, min)
  = (if listp (nodes)
      then min (node-value (car (nodes), state),
             min-node-value (cdr (nodes), state, min))
      else min endif

DEFINITION:
correct (tree, state)
  = (found-value (car (tree), state)
      = min-node-value (cdr (nodes (tree)),
                       state,
                       node-value (car (tree), state)))
;;; Proof of Correctness

**Theorem: all-empty-implies-empty**

\[(\text{all-empty}(\text{channels}, \text{state}) \land (\text{channel} \in \text{channels})) \rightarrow (\neg \text{listp}(\text{channel}(\text{channel}, \text{state})))\]

**Theorem: not-started-implies-not-started**

\[(\text{not-started}(\text{nodes}, \text{state}) \land (\text{node} \in \text{nodes})) \rightarrow (\text{cdr}(\text{assoc}(\text{cons}('\text{status}, \text{node}), \text{state})) = '\text{not-started})\]

**Theorem: all-empty-append**

\[(\text{all-empty}(\text{append}(a, b), \text{state}) = (\text{all-empty}(a, \text{state}) \land \text{all-empty}(b, \text{state}))\]

**Theorem: all-empty-implies-ul**

\[(\text{all-empty}(\text{up-links}, \text{state}) \rightarrow \text{ul}(\text{up-links}, \text{state})\]

**Definition:**

\[
\text{nodes-in-channels}(\text{channels}) = \text{if listp}(\text{channels}) \text{ then cons(caar(\text{channels}), )} \text{ cons(cdar(\text{channels}), nodes-in-channels(cdr(\text{channels})))) else nil endif}
\]

**Theorem: all-empty-not-started-implies-dl**

\[(\text{all-empty}(\text{down-links}, \text{state}) \land \text{not-started}(\text{nodes-in-channels}(\text{down-links}, \text{state})) \rightarrow \text{dl}(\text{down-links}, \text{state})\]

**Theorem: not-started-implies-no**

\[(\text{not-started}(\text{nodes}, \text{state}) \rightarrow \text{no}(\text{nodes}, \text{tree}, \text{state})\]

**Theorem: nodes-in-down-links-1-in-nodes**

\[(\text{node} \in \text{nodes-in-channels}(\text{down-links-1}(\text{parent}, \text{children}))) = \text{if listp}(\text{children}) \text{ then node} \in \text{cons}(\text{parent}, \text{children}) \text{ else f endif}\]

**Theorem: nodes-in-channels-append**

\[\text{nodes-in-channels}(\text{append}(a, b)) = \text{append}(\text{nodes-in-channels}(a), \text{nodes-in-channels}(b))\]

**Theorem: nodes-in-down-links-in-nodes**

\[(\text{proper-tree}('\text{tree}, \text{tree}) \land (\text{node} \in \text{nodes-in-channels}(\text{down-links}(\text{nodes}, \text{tree})))) \rightarrow (\text{node} \in \text{nodes}(\text{tree}))\]

19
THEOREM: sublistp-not-started
(sublistp (sub, list) ∧ not-started (list, state)) → not-started (sub, state)

THEOREM: sublistp-down-links-1
(sublistp (children, nodes) ∧ (parent ∈ nodes))
→ sublistp (nodes-in-channels (down-links-1 (parent, children)), nodes)

THEOREM: children-of-non-node
(parent ∉ nodes-rec (flag, tree))
→ (children-rec (flag, parent, tree) = nil)

THEOREM: down-links-is-sublistp
proper-tree (tree)
→ sublistp (nodes-in-channels (down-links (nodes, tree)),
            nodes-rec (tree, tree))

THEOREM: initial-conditions-imply-invariant
(proper-tree (tree)
∧ all-empty (all-channels (tree), state)
∧ not-started (nodes (tree), state))
→ inv (tree, state)

DEFINITION:
found-value-node-value (subtrees, state)
= if listp (subtrees)
   then (found-value (caar (subtrees), state)
      = min-node-value (cdr (nodes-rec (tree, car (subtrees)))).
   node-value (caar (subtrees), state))
   ∧ found-value-node-value (cdr (subtrees), state)
   else t endif

DEFINITION:
nati (subtrees)
= if listp (subtrees) then nati (next-level (subtrees))
else t endif

THEOREM: found-value-node-value-append
found-value-node-value-append (a, b, state)
= (found-value-node-value (a, state) ∧ found-value-node-value (b, state))

;find-value-of-node-value for a subtree is true if
;find-value-of-node-value for the next-level of that subtree is true.
Theorem: no-implies
(no (nodes, tree, state)
∧ (node ∈ nodes)
∧ (status (node, state) = 'started))
→ ((number-not-reported (children (node, tree), node, state)
    = cdr (assoc (cons ('outstanding, node), state)))
∧ (cdr (assoc (cons ('outstanding, node), state)) ∈ N)
∧ (cdr (assoc (cons ('found-value, node), state))
    = min-of-reported (children (node, tree),
                        node, state,
                        node-value (node, state))))

Theorem: total-outstanding-0-implies
((total-outstanding (nodes, tree, state) = 0)
∧ (node ∈ nodes)
∧ (cdr (assoc (cons ('outstanding, node), state)) ∈ N))
→ (((total-outstanding (nodes, tree, state) = 0) ∧ (node ∈ nodes))
    → (cdr (assoc (cons ('status, node), state)) = 'started))

Theorem: number-not-reported-0-implies
((number-not-reported (children, parent, state) = 0)
∧ (node ∈ children))
→ reported (node, parent, state)

Theorem: proper-tree-tree-implies-nodes-exists
proper-tree (tree, tree) → listp (nodes-rec (tree, tree))

Theorem: min-of-two-nodes-values
min (min-node-value (forest-1, state),
    cdr (assoc (cons ('node-value, root), state)))
∧ min-node-value (rest-of-forest, state, min)
= min-node-value (cons (root, append (forest-1, rest-of-forest)), state, min)

Theorem: found-value-min-value-generalized
(found-value-node-value (forest, state)
∧ (number-not-reported (roots (forest), root, state) = 0)
∧ proper-tree (forest, forest))
→ (min-of-reported (roots (forest), root, state, min)
    = min-node-value (nodes-rec (forest, forest), state, min))

Theorem: no-at-termination
(proper-tree (tree, tree)
∧ proper-tree (forest, subtrees)
\[\begin{align*}
&\land \text{setp} (\text{nodes-rec} (\text{tree}, \text{tree})) \\
&\land \text{no} (\text{nodes-rec} (\text{tree}, \text{tree}, \text{state})) \\
&\land (\text{total-outstanding} (\text{nodes-rec} (\text{tree}, \text{tree}, \text{state}) = 0)) \\
&\land \text{sublistp} (\text{subtrees}, \text{subtrees} (\text{tree}, \text{tree}))) \\
&\rightarrow \text{found-value-node-value} (\text{subtrees}, \text{state})
\end{align*}\]

**Theorem:** inv-implies-augmented-correctness-condition

(proper-tree (tree, tree)
\land \text{setp} (\text{nodes-rec} (\text{tree}, \text{tree}))
\land \text{inv} (\text{tree}, \text{state})
\land (\text{total-outstanding} (\text{nodes} (\text{tree}, \text{tree}, \text{state}) = 0))
\rightarrow \text{correct} (\text{tree}, \text{state})

**Definition:**

\[\begin{align*}
\text{send-find-func} (\text{to-children}, \text{old}) = &\ 
\text{if} \ \text{listp} (\text{to-children}) \\
&\text{then} \ \text{update-assoc} (\text{car} (\text{to-children}), \ \\
&\text{send} (\text{car} (\text{to-children}), \ '\text{find}, \text{old}), \ \\
&\text{send-find-func} (\text{cdr} (\text{to-children}), \text{old})) \\
&\text{else} \ \text{old} \ \text{endif}
\end{align*}\]

**Definition:**

\[\begin{align*}
\text{receive-find-func} (\text{old}, \text{node}, \text{from-parent}, \text{to-parent}, \text{to-children}) = &\ 
\text{if} \ \text{head} (\text{from-parent}, \text{old}) = '\text{find} \\
&\text{then} \ \text{update-assoc} (\text{from-parent}, \ \\
&\text{receive} (\text{from-parent}, \text{old}), \ \\
&\text{update-assoc} (\text{cons} (\text{'status}, \text{node}), \ \\
&'\text{started}, \ \\
&\text{update-assoc} (\text{cons} (\text{'found-value}, \text{node}), \ \\
&\text{node-value} (\text{node}, \text{old}), \ \\
&\text{update-assoc} (\text{cons} (\text{'outstanding}, \text{node}), \ \\
&\text{length} (\text{to-children}), \ \\
&\text{if} \ \text{length} (\text{to-children}) \simeq 0 \\
&\text{then} \ \text{update-assoc} (\text{to-parent}, \ \\
&\text{send} (\text{to-parent}, \text{node-value} (\text{node}, \text{old}), \ \\
&\text{old}), \ \\
&\text{send-find-func} (\text{to-children}, \text{old})) \\
&\text{else} \ \text{send-find-func} (\text{to-children}, \text{old}) \ \text{endif})
\end{align*}\]

else \ text{old} endif

Theorem: send-find-func-implements-send-find
send-find (to-children, old, send-find-func (to-children, old))

Theorem: nodes-are-not-litatoms
(all-numberps (nodes-rec (flag, tree)) ∧ (node ∈ nodes-rec (flag, tree)))
→ ((pack (x) = node) = f)

Theorem: parent-is-not-a-litatom
(all-numberps (nodes-rec ('tree, tree))
∧ setp (nodes-rec ('tree, tree))
∧ proper-tree ('tree, tree)
∧ (child ∈ cdr (nodes-rec ('tree, tree))))
→ ((pack (x) = car (parent-rec ('tree, child, tree))) = f)

Theorem: children-are-not-litatoms
(all-numberps (nodes-rec (flag, tree))
∧ proper-tree (flag, tree)
∧ (child ∈ children-rec (flag, parent, tree)))
→ ((pack (x) = child) = f)

Theorem: children-are-not-litatoms-member
(all-numberps (nodes-rec (flag, tree)) ∧ proper-tree (flag, tree))
→ ((pack (x) ∈ children-rec (flag, parent, tree)) = f)

Theorem: send-find-of-update-assoc
(key ̸∈ to-children)
→ (send-find (to-children, old, update-assoc (key, value, state)) = send-find (to-children, old, state))

Theorem: assoc-of-send-find-func
(key ̸∈ to-children)
→ (assoc (key, send-find-func (to-children, old)) = assoc (key, old))

Theorem: about-rfp
(p ̸∈ c) → (cons (v, p) ̸∈ rfp (v, c))

Theorem: about-rfp-numberp
(a ∈ N) → (cons (pack (x), y) ̸∈ rfp (a, b))

Theorem: parent-not-in-rfp
(setp (nodes-rec ('tree, tree))
∧ proper-tree ('tree, tree)
∧ (v ∈ cdr (nodes-rec ('tree, tree))))
→ (cons (v, car (parent-rec ('tree, v, tree))) ̸∈ rfp (v, children-rec ('tree, v, tree)))
Theorem: to-node-not-in-rfp

\((node \not\in children) \rightarrow (\text{cons}(x, node) \not\in \text{rfp}(node, children))\)

Theorem: uc-of-send-find-func

\[\text{sublistp}(\text{to-children}, \text{except}) \rightarrow (\text{uc}(old, \text{send-find-func}(\text{to-children}, state), keys, except) = \text{uc}(old, state, keys, except))\]

Theorem: receive-find-func-implements-receive-find

\((\text{treep}(tree) \land (statement \in \text{receive-find-prg}(\text{cdr}(\text{nodes}(tree)), tree))) \rightarrow \text{n}(old, \text{receive-find-func}(old, cadr(statement), caddr(statement), cadddr(statement), cadddd'r(statement)), state)\)

Definition:

receive-report-func \((old, node, from-child, to-parent)\) = if empty \((from-child, old)\) then \(old\) else update-assoc \((from-child, old)\), receive \((from-child, old)\), update-assoc \((\text{cons}(\text{'found-value}, node), \text{min}(\text{found-value}(node, old), \text{head}(\text{from-child}, old)))\), update-assoc \((\text{cons}(\text{'outstanding}, node), \text{outstanding}(node, old) - 1)\), if \((\text{outstanding}(node, old) - 1) \approx 0\) then update-assoc \((\text{to-parent}, \text{send}(\text{to-parent}, \text{min}(\text{found-value}(node, old), \text{head}(\text{from-child}, old)), old)))\) else \(old\) endif

Theorem: receive-report-func-implements-receive-report

\((\text{treep}(tree) \land (statement \in \text{receive-report-prg}(\text{cdr}(\text{nodes}(tree)), tree))) \rightarrow \text{n}(old,\)
receive-report-func (old, 
cadr (statement),  
caddr (statement),  
caddr (statement)),

statement)

**Definition:**
start-func (old, root, to-children)
= if status (root, old) = not-started
then update-assoc (cons ('status, root), 
'started, 
update-assoc (cons ('found-value, root), 
node-value (root, old), 
update-assoc (cons ('outstanding, root), 
length (to-children), 
send-find-func (to-children, old)))
else old endif

**Theorem:** start-func-implements-start
(treep (tree) ∧ (statement ∈ start-prg (car (tree), tree)))
→ n (old, start-func (old, cadr (statement), caddr (statement)), statement)

**Definition:**
root-receive-report-func (old, root, from-child)
= if empty (from-child, old) then old
else update-assoc (from-child, 
receive (from-child, old),
update-assoc (cons ('found-value, root), 
min (found-value (root, old), 
head (from-child, old)),
update-assoc (cons ('outstanding, root),
outstanding (root, old) − 1, 
old))) endif

**Theorem:** root-receive-report-func-implements-root-receive-report
(treep (tree) ∧ (statement ∈ root-receive-report-prg (car (tree), tree)))
→ n (old, 
root-receive-report-func (old, cadr (statement), caddr (statement)), statement)

**Theorem:** receive-find-prg-is-total
treep (tree)  
→  total-sufficient (statement,  
    receive-find-prg (cdr (nodes (tree)), tree),  
    old,  
    receive-find-func (old,  
        cadr (statement),  
        caddr (statement),  
        cadddr (statement),  
        caddddr (statement)))

THEOREM: receive-report-prg-is-total  
treep (tree)  
→  total-sufficient (statement,  
    receive-report-prg (cdr (nodes (tree)), tree),  
    old,  
    receive-report-func (old,  
        cadr (statement),  
        caddr (statement),  
        cadddr (statement)))

THEOREM: start-prg-is-total  
treep (tree)  
→  total-sufficient (statement,  
    start-prg (car (tree), tree),  
    old,  
    start-func (old, cadr (statement), caddr (statement)))

THEOREM: root-receive-report-prg-is-total  
treep (tree)  
→  total-sufficient (statement,  
    root-receive-report-prg (car (tree), tree),  
    old,  
    root-receive-report-func (old,  
        cadr (statement),  
        caddr (statement)))

THEOREM: total-tree-prg  
treep (tree) → total (tree-prg (tree))

THEOREM: listp-tree-prg  
listp (tree-prg (tree))

THEOREM: node-values-constant-unless-sufficient  
(treep (tree) ∧ (node ∈ nodes (tree)))  
→  unless-sufficient (statement,  

tree-prg(tree),
old,
new,
'(equal (node-value ',node state) ',k),
'(false))

**Theorem**: node-values-constant-invariant
(initial-condition ('(and
(all-empty ',(all-channels tree) state)
(and
(not-started ',(nodes tree) state)
(equal (node-value ',node state) ',k))),
tree-prg(tree))
∧ treep(tree)
∧ (node ∈ nodes(tree)))
→ invariant(''(equal (node-value ',node state) ',k),
tree-prg(tree))

**Theorem**: dl-implies-instance-of-dl
(dl (down-links, state) ∧ (down-link ∈ down-links))
→ ((empty (down-link, state)
∧ (status (car (down-link), state)
   = status (cdr (down-link), state)))
∨ ((channel (down-link, state) = list ('find)
   ∧ (status (car (down-link), state) = 'started)
   ∧ (status (cdr (down-link), state) = 'not-started)))

**Event**: Disable dl-implies-instance-of-dl.

**Theorem**: ul-implies-instance-of-ul
(ul (uplinks, state) ∧ (uplink ∈ uplinks))
→ (empty (uplink, state)
∨ ((channel (uplink, state)
   = list (found-value (car (uplink), state))
   ∧ done (car (uplink), state)))

**Event**: Disable ul-implies-instance-of-ul.

**Theorem**: ul-implies-instance-of-ul-not-empty-uplink
(ul (uplinks, state) ∧ (uplink ∈ uplinks) ∧ (¬ empty (uplink, state)))
→ ((cdr (assoc (uplink, state)) = list (found-value (car (uplink), state)))
∧ (cdr (assoc (cons ('status, car (uplink)), state))
   = 'started)
∧ (cdr (assoc (cons ('outstanding, car (uplink)), state)) ≃ 0))
Theorem: no-implies-instance-of-no
(no (nodes, tree, state)
  \land (node \in nodes)
  \land (status (node, state) = 'started)
  \rightarrow ((cdr (assoc (cons ('outstanding, node), state)))
      = \text{number-not-reported}(\text{children-rec}('tree, node, tree),
       node, state))
  \land (cdr (assoc (cons ('found-value, node), state)))
      = \text{min-of-reported}(\text{children-rec}('tree, node, tree),
       node, state, \text{node-value}(node, state))))

Theorem: member-down-links-1
(down-link \in down-links-1 (parent, children))
= ((\text{car} (down-link) = parent)
  \land (\text{cdr} (down-link) \in children)
  \land \text{listp}(down-link))

Theorem: member-down-links
(down-link \in down-links (nodes, tree))
= ((\text{car} (down-link) \in nodes)
  \land (\text{cdr} (down-link) \in children (\text{car} (down-link), tree))
  \land \text{listp}(down-link))

Theorem: parent-not-child
(proper-tree (flag, tree) \land \text{setp}(\text{nodes-rec} (flag, tree)))
\rightarrow (parent \not\in \text{children-rec} (flag, parent, tree))

Theorem: parent-not-grandchild
(proper-tree (flag, tree)
  \land \text{setp}(\text{nodes-rec} (flag, tree))
  \land (child \in \text{children-rec} (flag, parent, tree)))
\rightarrow (parent \not\in \text{children-rec} (flag, child, tree))

Theorem: parent-of-parent-not-node
(proper-tree (flag, tree)
  \land \text{setp}(\text{nodes-rec} (flag, tree))
  \land \text{listp}(\text{parent-rec} (flag, node, tree))
  \land \text{listp}(\text{parent-rec} (flag, \text{car} (\text{parent-rec} (flag, node, tree)), tree))
\rightarrow (\text{car} (\text{parent-rec} (flag, \text{car} (\text{parent-rec} (flag, node, tree)), tree)) \neq node))

Theorem: member-rfp
(channel \in rfp (parent, children))
\[ ((\text{car} (\text{channel}) = \text{parent}) \land (\text{cdr} (\text{channel}) \in \text{children}) \land \text{listp} (\text{channel})) \]

**Theorem:** send-find-implies
\[ (\text{send-find} (\text{channels, old, new}) \land (\text{key} \in \text{channels})) \rightarrow (\text{cdr} (\text{assoc} (\text{key, new})) = \text{send} (\text{key, 'find, old})) \]

**Theorem:** assoc-of-channel-preserved-root-receive-report
\[ ((w \notin \text{nodes-rec} (\text{'forest, d})) \land \text{setp} (\text{nodes-rec} (\text{'forest, d})) \land (z \in \text{nodes-rec} (\text{'forest, d})) \land \text{uc} (\text{new, old, append (strip-cars (new), strip-cars (old)), list (cons (v, w), cons (\text{'outstanding, w}), cons (\text{'found-value, w}))})) \rightarrow (\text{assoc} (\text{cons (x, z), new}) = \text{assoc} (\text{cons (x, z), old})) \]

**Theorem:** assoc-equal-cons
\[ (\text{assoc} (\text{key, alist}) = \text{cons (key, value)}) = (\text{listp} (\text{assoc} (\text{key, alist})) \land (\text{cdr} (\text{assoc} (\text{key, alist})) = \text{value})) \]

**Theorem:** send-find-general
\[ (\text{send-find} (\text{channels, old, new}) \land (\text{key} \in \text{channels})) \rightarrow (\text{assoc} (\text{key, new}) = \text{cons (key, send (key, 'find, old)))) \]

**Theorem:** all-numberps-do-not-contain-litatom
\[ \text{all-numberps} (\text{list}) \rightarrow (\text{pack} (x) \notin \text{list}) \]

**Theorem:** all-numberps-append
\[ \text{all-numberps} (\text{append} (x, y)) = (\text{all-numberps} (x) \land \text{all-numberps} (y)) \]

**Theorem:** all-numberps-nodes-implies-all-numberps-parent
\[ \text{all-numberps} (\text{nodes-rec} (\text{flag, tree})) \rightarrow \text{all-numberps} (\text{parent-rec} (\text{flag, child, tree})) \]

**Theorem:** all-numberps-nodes-implies-all-numberps-car-parent
\[ \text{all-numberps} (\text{nodes-rec} (\text{flag, tree})) \rightarrow (\text{car} (\text{parent-rec} (\text{flag, child, tree})) \in \mathbb{N}) \]

**Theorem:** parent-not-litatom
\[ \text{all-numberps} (\text{nodes-rec} (\text{flag, tree})) \rightarrow ((\text{pack} (x) = \text{car} (\text{parent-rec} (\text{flag, child, tree}))) = \text{f}) \]

**Theorem:** all-numberps-forest-implies-all-numberps-roots
\[ \text{all-numberps} (\text{nodes-rec} (\text{'forest, forest})) \rightarrow \text{all-numberps} (\text{roots} (\text{forest})) \]
Theorem: all-numberps-nodes-implies-all-numberps-children
\[
\text{all-numberps (nodes-rec (flag, tree))} \\
\rightarrow \text{all-numberps (children-rec (flag, parent, tree))}
\]

Theorem: dl-preserves-instance-of-dl
\[
\text{(treep (tree)} \\
\land (down-link \in \text{down-links (nodes (tree), tree)} \\
\land \text{n (old, new, statement)} \\
\land (\text{statement }\in \text{tree-prg (tree)}) \\
\land \text{dl (down-links (nodes (tree), tree), old)}) \\
\rightarrow ((\text{empty (down-link, new)} \\
\land (\text{status (car (down-link), new)} = \text{status (cdr (down-link), new)})) \\
\lor ((\text{channel (down-link, new)} = \text{list ('find)} \\
\land (\text{status (car (down-link), new)} = '\text{started}) \\
\land (\text{status (cdr (down-link), new)} = '\text{not-started})))
\]

Theorem: dl-preserves-sublist
\[
\text{(dl (down-links (nodes (tree), tree), old)} \\
\land \text{treep (tree)} \\
\land \text{n (old, new, statement)} \\
\land (\text{statement }\in \text{tree-prg (tree)}) \\
\land \text{sublistp (sublist, down-links (nodes (tree), tree))}) \\
\rightarrow \text{dl (sublist, new)}
\]

Theorem: dl-preserves-dl
\[
\text{(dl (down-links (nodes (tree), tree), old)} \\
\land \text{treep (tree)} \\
\land \text{n (old, new, statement)} \\
\land (\text{statement }\in \text{tree-prg (tree)}) \\
\rightarrow \text{dl (down-links (nodes (tree), tree), new)}
\]

Theorem: member-up-links
\[
(\text{up-link }\in \text{up-links (nodes, tree)}) = ((\text{car (up-link)} \in \text{nodes)} \\
\land (\text{cdr (up-link)} = \text{parent (car (up-link), tree)} \\
\land \text{listp (up-link)})
\]

Theorem: zero-not-reported-implies-children-reported
\[
((\text{number-not-reported (children, parent, state) }\simeq 0) \land (\text{child }\in \text{children})) \\
\rightarrow ((\text{cdr (assoc (cons ('status, child), state)) }= '\text{started}) \\
\land (\text{outstanding (child, state) }\simeq 0) \\
\land (\neg \text{listp (cdr (assoc (cons (child, parent), state))}))
\]

Theorem: dl-ul-no-preserves-instance-of-ul
\[
\text{(treep (tree)}
\]

30
∧ (up-link ∈ up-links (cdr (nodes (tree)), tree))
∧ n (old, new, statement)
∧ (statement ∈ tree-prg (tree))
∧ dl (down-links (nodes (tree), tree), old)
∧ ul (up-links (cdr (nodes (tree)), tree), old)
∧ no (nodes (tree), tree, old))
→ (empty (up-link, new)
    ∨ ((channel (up-link, new)
        = list (found-value (car (up-link), new)))
      ∧ done (car (up-link), new)))

THEOREM: dl-ul-no-preserves-ul-sublist
(treep (tree)
 ∧ n (old, new, statement)
 ∧ (statement ∈ tree-prg (tree))
 ∧ dl (down-links (nodes (tree), tree), old)
 ∧ ul (up-links (cdr (nodes (tree)), tree), old)
 ∧ no (nodes (tree), tree, old)
 ∧ sublistp (sublist, up-links (cdr (nodes (tree)), tree)))
→ ul (sublist, new)

THEOREM: dl-ul-no-preserves-ul
(treep (tree)
 ∧ n (old, new, statement)
 ∧ (statement ∈ tree-prg (tree))
 ∧ dl (down-links (nodes (tree), tree), old)
 ∧ ul (up-links (cdr (nodes (tree)), tree), old)
 ∧ no (nodes (tree), tree, old)
 → ul (up-links (cdr (nodes (tree)), tree), new)

THEOREM: parent-not-started-implies-all-empty-and-not-started
((status (parent, state) = 'not-started)
 ∧ dl (rfp (parent, children), state))
→ (all-empty (rfp (parent, children), state)
    ∧ not-started (children, state))

THEOREM: start-preserves-no-for-parent
((parent ∈ N)
 ∧ (parent ∉ children)
 ∧ not-started (children, old)
 ∧ sublistp (rfp (parent, children), rfp (parent, except))
 ∧ changed (old, new,
             append (list (cons ('status, parent),
                             cons ('found-value, parent),
                             children), old)))

31
\[\text{cons('outstanding, parent)},\]
\[\text{rfp(parent, except))}\]
\[\rightarrow (\text{number-not-reported(children, parent, new) = length(children)}\]
\[\quad \land (\text{min-of-reported(children, parent, new, value) = value})\]

**Theorem: unchanged-preserves-no**

\[
\text{changed(old, new, nil)}
\rightarrow (\text{number-not-reported(children, parent, new)}
\quad = \text{number-not-reported(children, parent, old})
\quad \land (\text{min-of-reported(children, parent, new, value)}
\quad = \text{min-of-reported(children, parent, old, value}))\]

**Theorem: start-preserves-no-for-rest-of-tree**

\[
((\text{root} \in \mathbb{N})
\quad \land (\text{parent} \in \mathbb{N})
\quad \land (\text{parent} \notin \text{children})
\quad \land (\text{root} \notin \text{children})
\land (\text{root} \neq \text{parent})
\quad \land \text{changed(old, new, append(list(cons('status, root),
\text{cons('found-value, root),
\text{cons('outstanding, root)),
\text{rfp(root, except))})})}
\rightarrow (\text{number-not-reported(children, parent, new)}
\quad = \text{number-not-reported(children, parent, old})
\quad \land (\text{min-of-reported(children, parent, new, value)}
\quad = \text{min-of-reported(children, parent, old, value}))\]

**Theorem: length-rfp**

\[
\text{length(rfp(parent, children)) = length(children)}\]

**Theorem: start-preserves-instance-of-no**

\[
(\text{treep(tree)}
\quad \land \text{start(old, new, car(tree), rfp(car(tree), children(car(tree), tree)))}
\land (\text{node} \in \text{nodes(tree)})
\land (\text{dl(rfp(car(tree), children(car(tree), tree)), old})
\land (\text{status(node, new) = 'started})
\land ((\text{status(node, old) = 'started})
\rightarrow ((\text{outstanding(node, old)}
\quad = \text{number-not-reported(children(node, tree), node, old)})
\land (\text{found-value(node, old)}
\quad = \text{min-of-reported(children(node, tree),
\text{node, old)}})}
Theorem: min-commutative
\[ \min(a, b) = \min(b, a) \]

Theorem: min-associative
\[ \min(\min(a, b), c) = \min(a, \min(b, c)) \]

Theorem: min-commutative-1
\[ \min(a, \min(b, c)) = \min(b, \min(a, c)) \]

Theorem: min-of-reported-of-min
\[ \min-of-reported(children, parent, state, \min(value, x)) = \min(\min-of-reported(children, parent, state, value), x) \]

Theorem: update-min-of-reported
\[ (parent \in N) \]
\[ \land (child \in N) \]
\[ \land (parent \neq child) \]
\[ \land \text{all-numberps}(children) \]
\[ \land \text{setp}(children) \]
\[ \land (parent \notin children) \]
\[ \land (\text{channel}(\text{cons}(child, parent), old) = \text{list}(\text{found-value}(child, old))) \]
\[ \land \text{done}(child, old) \]
\[ \land (\text{channel}(\text{cons}(child, parent), new) = \text{receive}(\text{cons}(child, parent), old)) \]
\[ \land \text{changed}(old, \]
\[ \land \text{new,} \]
\[ \land \text{cons}(\text{\textit{outstanding}}, parent), \]
\[ \land \text{cons}(\text{\textit{found-value}}, parent))))) \]
\[ \rightarrow (\min-of-reported(children, parent, new, value) \]
\[ = \text{if } child \in children \]
\[ \text{then} \min(\text{found-value}(child, old), \]
\[ \min-of-reported(children, parent, old, value)) \]
\[ \text{else} \min-of-reported(children, parent, old, value) \text{endif}) \]

Theorem: min-of-reported-of-non-root
\[ ((root \in N) \]

33
\( \land (\text{child} \in \mathbb{N}) \)
\( \land (\text{parent} \in \mathbb{N}) \)
\( \land \text{all-numberps} (\text{children}) \)
\( \land \text{setp} (\text{children}) \)
\( \land (\text{parent} \notin \text{children}) \)
\( \land (\text{root} \neq \text{parent}) \)
\( \land (\text{root} \notin \text{children}) \)
\( \land \text{changed} (\text{old}, \text{new}, \text{list} (\text{cons} (\text{child}, \text{root}), \text{cons} (\text{’outstanding}, \text{root}), \text{cons} (\text{’found-value}, \text{root})))) \)
\( \rightarrow \text{min-of-reported} (\text{children}, \text{parent}, \text{new}, \text{value}) = \text{min-of-reported} (\text{children}, \text{parent}, \text{old}, \text{value}) \)

**Theorem: number-not-reported-of-non-root**

\((\text{root} \in \mathbb{N}) \land (\text{child} \in \mathbb{N}) \land (\text{parent} \in \mathbb{N}) \land \text{all-numberps} (\text{children}) \land \text{setp} (\text{children}) \land (\text{parent} \notin \text{children}) \land (\text{root} \neq \text{parent}) \land (\text{root} \notin \text{children}) \land \text{changed} (\text{old}, \text{new}, \text{list} (\text{cons} (\text{child}, \text{root}), \text{cons} (\text{’outstanding}, \text{root}), \text{cons} (\text{’found-value}, \text{root})))) \rightarrow \text{number-not-reported} (\text{children}, \text{parent}, \text{new}) = \text{number-not-reported} (\text{children}, \text{parent}, \text{old}) \)

**Theorem: number-not-reported-of-root**

\((\text{parent} \in \mathbb{N}) \land (\text{child} \in \mathbb{N}) \land (\text{parent} \neq \text{child}) \land \text{all-numberps} (\text{children}) \land \text{setp} (\text{children}) \land (\text{parent} \notin \text{children}) \land (\text{channel} (\text{cons} (\text{child}, \text{parent}), \text{old}) = \text{list} (\text{found-value} (\text{child}, \text{old}))) \land \text{done} (\text{child}, \text{old}) \land (\text{channel} (\text{cons} (\text{child}, \text{parent}), \text{new}) = \text{receive} (\text{cons} (\text{child}, \text{parent}), \text{old})) \land \text{changed} (\text{old}, \text{new}, \text{list} (\text{cons} (\text{child}, \text{root}), \text{cons} (\text{’outstanding}, \text{root}), \text{cons} (\text{’found-value}, \text{root})))) \rightarrow \text{number-not-reported} (\text{children}, \text{parent}, \text{new}) = \text{number-not-reported} (\text{children}, \text{parent}, \text{old}) \)
Theorem: setp-nodes-implies-setp-roots
(proper-tree ('forest, forest) ∧ setp (nodes-rec ('forest, forest)))
→ setp (roots (forest))

Theorem: setp-nodes-setp-children
(proper-tree (flag, tree) ∧ setp (nodes-rec (flag, tree)))
→ setp (children-rec (flag, parent, tree))

Theorem: root-receive-report-preserves-instance-of-no
(treep (tree)
∧ (child ∈ children (car (tree), tree))
∧ root-receive-report (old, new, car (tree), cons (child, car (tree)))
∧ ul (up-links (cdr (nodes (tree)), tree), old)
∧ (node ∈ nodes (tree))
∧ (status (node, new) = 'started)
∧ ((status (node, old) = 'started)
→ ((outstanding (node, old)
   = number-not-reported (children (node, tree), node, old))
   ∧ (found-value (node, old)
     = min-of-reported (children (node, tree),
                       node, old,
                       node-value (node, old))))))
→ ((outstanding (node, new)
   = number-not-reported (children (node, tree), node, new))
   ∧ (found-value (node, new)
     = min-of-reported (children (node, tree),
                       node, new,
                       node-value (node, new))))

Theorem: receive-find-preserves-no-for-rest-of-tree
((node ∈ N)
∧ (parent-of-node ∈ N)
∧ (parent ∈ N)
∧ (parent ≠ node)
∧ (node ∉ children)
∧ changed (old, new, append (list (cons (parent-of-node, node), cons (node, parent-of-node), cons (*status, node), cons (*found-value, node), cons (*outstanding, node)), rfp (node, except)))))
→ ((number-not-reported (children, parent, new) = number-not-reported (children, parent, old)) ∧ (min-of-reported (children, parent, new, value) = min-of-reported (children, parent, old, value)))

**Theorem:** receive-find-preserves-no-for-node

((node ∈ N) ∧ (parent-of-node ∈ N) ∧ (node ≠ children) ∧ not-started (children, old) ∧ sublistp (rfp (node, children), rfp (node, except)) ∧ changed (old, new, append (list (cons (parent-of-node, node), cons (node, parent-of-node), cons (*status, node), cons (*found-value, node), cons (*outstanding, node)), rfp (node, except)))))
→ ((number-not-reported (children, node, new) = length (children)) ∧ (min-of-reported (children, node, new, value) = min-of-reported (children, node, old, value)))

**Theorem:** receive-find-preserves-no-for-parent-of-node

((node ∈ N) ∧ (parent-of-node ∈ N) ∧ (node ≠ parent-of-node) ∧ (status (node, old) ≠ 'started) ∧ ((outstanding (node, new) ≃ 0) → (¬ empty (cons (node, parent-of-node), new)))) ∧ changed (old, new, append (list (cons (parent-of-node, node), cons (node, parent-of-node), cons (*status, node), cons (*found-value, node), cons (*outstanding, node)), rfp (node, except)))))
→ ((number-not-reported (children, node, new) = length (children)) ∧ (min-of-reported (children, node, new, value) = min-of-reported (children, node, old, value)))
cons('outstanding, node),
rfp(node, except)))
→ ((number-not-reported (children, parent-of-node, new)
  = number-not-reported (children, parent-of-node, old))
  ∧ (min-of-reported (children, parent-of-node, new, value)
  = min-of-reported (children, parent-of-node, old, value)))

Theorem: dl-of-append
dl (append (a, b), state) = (dl (a, state) ∧ dl (b, state))

Theorem: down-links-1-rfp
down-links-1 (parent, children) = rfp (parent, children)

Theorem: dl-down-links-implies-dl-rfp
(dl (down-links (nodes, tree), state) ∧ (node ∈ nodes))
→ dl (rfp (node, children (node, tree)), state)

Event: Disable dl-down-links-implies-dl-rfp.

Event: Disable down-links-1-rfp.

Event: Disable dl-of-append.

Theorem: receive-find-preserves-instance-of-no
(treep (tree)
 ∧ (node ∈ cdr (nodes (tree)))))
∧ receive-find (old, new, node, cons (parent (node, tree), node),
  cons (node, parent (node, tree)), rfp (node, children (node, tree)))
∧ dl (down-links (nodes (tree), tree), old)
∧ (n ∈ nodes (tree))
∧ (status (n, new) = 'started)
∧ ((status (n, old) = 'started)
→ ((outstanding (n, old)
  = number-not-reported (children (n, tree), n, old))
  ∧ (found-value (n, old)
  = min-of-reported (children (n, tree), n, old, node-value (n, old))))))
\[\rightarrow (\text{outstanding}(n, \text{new}) = \text{number-not-reported} (\text{children}(n, \text{tree}), n, \text{new})) \]
\[\land \text{found-value}(n, \text{new}) = \text{min-of-reported} (\text{children}(n, \text{tree}), n, \text{new}, \text{node-value}(n, \text{new}))\]

**Theorem:** receive-report-preserves-no-for-rest-of-tree

\[((\text{node} \in \mathbb{N}) \land (\text{parent-of-node} \in \mathbb{N}) \land (\text{child-of-node} \in \mathbb{N}) \land (\text{parent} \in \mathbb{N}) \land (\text{parent} \neq \text{node}) \land (\text{node} \notin \text{children}) \land \text{changed}(\text{old}, \text{new}, \text{list}(\text{cons}(\text{child-of-node}, \text{node}), \text{cons}(\text{node}, \text{parent-of-node}) \text{cons}('\text{outstanding}, \text{node}), \text{cons}('\text{found-value}, \text{node}))) \land \text{done}(\text{child}, \text{old}) \land \text{channel}(\text{cons}(\text{child}, \text{node}), \text{new}) = \text{receive}(\text{cons}(\text{child}, \text{node}), \text{old})) \land \text{changed}(\text{old}, \text{new}, \text{list}(\text{cons}(\text{child}, \text{node}), \text{cons}(\text{node}, \text{parent}), \text{cons}('\text{outstanding}, \text{node}), \text{cons}('\text{found-value}, \text{node})))) \]

\[\rightarrow (\text{number-not-reported} (\text{children}, \text{parent}, \text{new}) = \text{number-not-reported} (\text{children}, \text{parent}, \text{old})) \land (\text{min-of-reported} (\text{children}, \text{parent}, \text{new}, \text{value}) = \text{min-of-reported} (\text{children}, \text{parent}, \text{old}, \text{value}))\]

**Theorem:** receive-report-preserves-no-for-node

\[((\text{node} \in \mathbb{N}) \land (\text{parent} \in \mathbb{N}) \land (\text{node} \notin \text{children}) \land \text{all-numberps}(\text{children}) \land \text{setp}(\text{children}) \land (\text{channel}(\text{cons}(\text{child}, \text{node}), \text{old}) = \text{list} (\text{found-value}(\text{child}, \text{old}))) \land \text{done}(\text{child}, \text{old}) \land (\text{channel}(\text{cons}(\text{child}, \text{node}), \text{new}) = \text{receive}(\text{cons}(\text{child}, \text{node}), \text{old})) \land \text{changed}(\text{old}, \text{new}, \text{list}(\text{cons}(\text{child}, \text{node}), \text{cons}(\text{node}, \text{parent}) \text{cons}('\text{outstanding}, \text{node}), \text{cons}('\text{found-value}, \text{node})))) \]

\[\rightarrow (\text{number-not-reported} (\text{children}, \text{node}, \text{new}) = \text{if} (\text{child} \in \text{children} \text{then} \text{number-not-reported} (\text{children}, \text{node}, \text{old}) - 1 \text{else} \text{number-not-reported} (\text{children}, \text{node}, \text{old}))\]
else number-not-reported \( (\text{children, node, old}) \) endif
\[\wedge \ (\text{min-of-reported} (\text{children, node, new, value}) \]
= if \( \text{child} \in \text{children} \)
then \( \text{min} (\text{found-value} (\text{child, old}), \)
\( \text{min-of-reported} (\text{children, node, old, value})) \)
else \( \text{min-of-reported} (\text{children, node, old, value}) \) endif
\]

**Theorem:** receive-report-preserves-no-for-parent
\( ((\text{node} \in \mathbb{N}) \)
\( \wedge (\text{parent} \in \mathbb{N}) \)
\( \wedge (\text{node} \neq \text{parent}) \)
\( \wedge ((\text{outstanding} (\text{node, new}) \simeq 0) \rightarrow (\neg \text{empty} (\text{cons} (\text{node, parent}), \text{new}))) \)
\( \wedge (\text{outstanding} (\text{node, old}) \neq 0) \)
\( \wedge \text{changed} (\text{old, new}, \)
\( \text{list} (\text{cons} (\text{child, node}), \)
\( \text{cons} (\text{node, parent}), \)
\( \text{cons} (\text{‘outstanding, node}), \)
\( \text{cons} (\text{‘found-value, node}))) \)
\( \rightarrow ((\text{number-not-reported} (\text{children, parent, new}) \)
\( = \text{number-not-reported} (\text{children, parent, old})) \)
\( \wedge (\text{min-of-reported} (\text{children, parent, new, value}) \)
\( = \text{min-of-reported} (\text{children, parent, old, value}))) \)

**Theorem:** child-member-cdr-nodes
\( (\text{proper-tree} (\text{‘tree, tree}) \)
\( \wedge \text{setp} (\text{nodes-rec} (\text{‘tree, tree})) \)
\( \wedge (\text{child} \in \text{children-rec} (\text{‘tree, node, tree}))) \)
\( \rightarrow (\text{child} \in \text{cdr} (\text{nodes-rec} (\text{‘tree, tree}))) \)

**Theorem:** receive-report-preserves-instance-of-no
\( (\text{treep} (\text{tree}) \)
\( \wedge (\text{node} \in \text{cdr} (\text{nodes} (\text{tree}))) \)
\( \wedge (\text{child} \in \text{children} (\text{node, tree})) \)
\( \wedge (n \in \text{nodes} (\text{tree}) \)
\( \wedge \text{receive-report} (\text{old, new, node}, \)
\( \text{cons} (\text{child, node}), \)
\( \text{cons} (\text{node, parent} (\text{node, tree}))) \)
\( \wedge (\text{status} (n, \text{new}) = \text{‘started}) \)
\( \wedge \text{ul} (\text{up-links} (\text{cdr} (\text{nodes} (\text{tree})), \text{tree}), \text{old}) \)
\( \wedge \text{no} (\text{nodes} (\text{tree}), \text{tree}, \text{old}) \)
\( \wedge \text{dl} (\text{down-links} (\text{nodes} (\text{tree}), \text{tree}), \text{old})) \)
\( \rightarrow ((\text{outstanding} (n, \text{new}) \neq \text{number-not-reported} (\text{children} (n, \text{tree}, n, \text{new}))) \)
\(\land (\text{found-value}\ (n,\ new)\\ = \text{min-of-reported}\ (\text{children}\ (n,\ tree),\\ \quad n,\ new,\\ \quad \text{node-value}\ (n,\ new)))\)

**Theorem:** dl-ul-no-preserves-instance-of-no
\[
\begin{align*}
\& \quad \text{treep}\ (tree) \\
\& \quad \land \ n\ (old,\ new,\ statement) \\
\& \quad \land \ (\text{statement} \in \text{tree-prg}\ (tree)) \\
\& \quad \land \ \text{dl}\ (\text{down-links}\ (\text{nodes}\ (tree),\ tree),\ old) \\
\& \quad \land \ \text{ul}\ (\text{up-links}\ (\text{cdr}\ (\text{nodes}\ (tree)),\ tree),\ old) \\
\& \quad \land \ \text{no}\ (\text{nodes}\ (tree),\ tree,\ old) \\
\& \quad \land \ (\text{node} \in \text{nodes}\ (tree)) \\
\& \quad \land \ (\text{status}\ (\text{node},\ new) = '\text{started}') \\
\rightarrow \quad ((\text{outstanding}\ (\text{node},\ new) \\
\quad = \text{number-not-reported}\ (\text{children}\ (\text{node},\ tree),\ node,\ new)) \\
\quad \land \ (\text{found-value}\ (\text{node},\ new)\\ \quad = \text{min-of-reported}\ (\text{children}\ (\text{node},\ tree),\\ \quad \quad \text{node},\ new,\\ \quad \quad \text{node-value}\ (\text{node},\ new)))))
\]

**Theorem:** dl-ul-no-preserves-no-sublist
\[
\begin{align*}
\& \quad \text{treep}\ (tree) \\
\& \quad \land \ n\ (old,\ new,\ statement) \\
\& \quad \land \ (\text{statement} \in \text{tree-prg}\ (tree)) \\
\& \quad \land \ \text{dl}\ (\text{down-links}\ (\text{nodes}\ (tree),\ tree),\ old) \\
\& \quad \land \ \text{ul}\ (\text{up-links}\ (\text{cdr}\ (\text{nodes}\ (tree)),\ tree),\ old) \\
\& \quad \land \ \text{no}\ (\text{nodes}\ (tree),\ tree,\ old) \\
\& \quad \land \ \text{sublistp}\ (\text{sublist},\ \text{nodes}\ (tree))) \\
\rightarrow \quad \text{no}\ (\text{sublist},\ tree,\ new)
\]

**Theorem:** inv-preserves-inv
\[
\begin{align*}
\& \quad \text{treep}\ (tree) \\
\& \quad \land \ n\ (old,\ new,\ statement) \\
\& \quad \land \ (\text{statement} \in \text{tree-prg}\ (tree)) \\
\& \quad \land \ \text{inv}\ (\text{tree},\ old) \\
\rightarrow \quad \text{inv}\ (\text{tree},\ new)
\]

**Theorem:** inv-is-invariant
\[
\begin{align*}
\& \quad \text{initial-condition}\ ('(\text{and} \\
\quad \quad (\text{all-empty } '((\text{all-channels}\ \text{tree})\ \text{state})) \\
\quad \quad (\text{not-started } '((\text{nodes}\ \text{tree})\ \text{state})), \\
\quad \quad \text{tree-prg}\ (\text{tree})))
\end{align*}
\]

40
∧ treep(tree)
→ invariant(‘(inv ',tree state), tree-prg(tree))

**Theorem:** outstanding-non-increasing
(treep(tree)
∧ (statement ∈ tree-prg(tree))
∧ n(old, new, statement)
∧ dl(down-links(nodes(tree), tree), old)
∧ (node ∈ nodes(tree)))
→ (if status(node, old) = ’started then outstanding(node, old)
else 1 + length(children(node, tree)) endif
    ≠ if status(node, new) = ’started
    then outstanding(node, new)
    else 1 + length(children(node, tree)) endif)

**Theorem:** total-outstanding-non-increasing-sublist
(treep(tree)
∧ (statement ∈ tree-prg(tree))
∧ n(old, new, statement)
∧ dl(down-links(nodes(tree), tree), old)
∧ sublistp(sublist, nodes(tree)))
→ (total-outstanding(sublist, tree, old)
    ≠ total-outstanding(sublist, tree, new))

**Theorem:** total-outstanding-non-increasing
(treep(tree)
∧ (statement ∈ tree-prg(tree))
∧ n(old, new, statement)
∧ dl(down-links(nodes(tree), tree), old))
→ (total-outstanding(nodes(tree), tree, old)
    ≠ total-outstanding(nodes(tree), tree, new))

**Theorem:** position-append
position(append(a, b), e) = if e ∈ a then position(a, e)
else length(a) + position(b, e) endif

**Theorem:** parents-position-decreases
((node ∈ nodes-rec(flag, tree))
∧ setp(nodes-rec(flag, tree))
∧ proper-tree(flag, tree)
∧ if flag = ’tree then car(tree) ≠ node
else node ∉ roots(tree) endif)
→ (position(nodes-rec(flag, tree), car(parent-rec(flag, node, tree)))
    < position(nodes-rec(flag, tree), node))
**Definition:**
parent-to-root-induction \((node, tree)\)
= if \((node \in \text{nodes}(tree))\)
  & \(\text{setp}(\text{nodes}(tree))\)
  & proper-tree\('tree, tree\)
then if \(\text{car}(tree) = node\) then \(t\)
  else parent-to-root-induction (parent \((node, tree)\), \(tree\)) endif
else \(t\) endif

**Theorem:** dl-and-all-empty-implies-root-defines-status
\(\text{dl}(\text{down-links}(\text{nodes}(tree), tree), state)\)
& all-empty \((\text{down-links}(\text{nodes}(tree), tree), state)\)
& \(\text{setp}(\text{nodes}(tree))\)
& proper-tree\('tree, tree\)
& \((node \in \text{nodes}(tree))\)
\(\rightarrow\) (cdr \((\text{assoc}(\text{cons}(\text{status}, node), state))\)) = \text{status}(\text{car}(tree), state))

**Definition:**
suffix \((s, l)\)
= if \(\text{listp}(l)\)
  then if \(s = l\) then \(t\)
    else suffix \((s, \text{cdr}(l))\) endif
  else \(\neg\ \text{listp}(s)\) endif

**Theorem:** suffix-implies-suffix-cdr
suffix \((s, l)\) \(\rightarrow\) suffix \((\text{cdr}(s), l)\)

**Theorem:** member-suffix-member-list
\((e \in s) \land \text{suffix}(s, l)\) \(\rightarrow\) \((e \in l)\)

**Theorem:** childs-position-increases
\((node \in \text{nodes-rec}(flag, tree))\)
& \(\text{setp}(\text{nodes-rec}(flag, tree))\)
& proper-tree\(\text{flag}, tree\)
& \((\text{child} \in \text{children-rec}(\text{flag}, node, tree))\)
\(\rightarrow\) (position \((\text{nodes-rec}(\text{flag}, tree), node)\)
  < position \((\text{nodes-rec}(\text{flag}, tree), \text{child})\))

**Theorem:** setp-list-setp-suffix
\((\text{setp}(l) \land \text{suffix}(s, l))\) \(\rightarrow\) \(\text{setp}(s)\)

**Theorem:** later-positions-are-in-suffix
\(\text{setp}(l)\)
& \(\text{suffix}(s, l)\)
& \((x \in s)\)

42
\( (y \in l) \land (\text{position}(l, x) < \text{position}(l, y)) \rightarrow (y \in s) \)

**Definition:**

\[
\text{all-done}(\text{nodes}, \text{state}) = \begin{cases} 
\text{if} & \text{listp}(\text{nodes}) \\
\text{then if} & \text{done}(\text{car}(\text{nodes}), \text{state}) \text{ then all-done}(\text{cdr}(\text{nodes}), \text{state}) \\
\text{else} & \text{f} \end{cases} \\
\text{else} & \text{t} \end{cases}
\]

**Theorem:** all-done-implies-done

\( (\text{all-done}(\text{nodes}, \text{state}) \land \text{node} \in \text{nodes}) \rightarrow \text{done}(\text{node}, \text{state}) \)

**Theorem:** all-done-implies-all-done-sublist

\( (\text{all-done}(\text{nodes}, \text{state}) \land \text{sublistp}(\text{sublist}, \text{nodes})) \rightarrow \text{all-done}(\text{sublist}, \text{state}) \)

**Definition:**

\[
\text{ulnks}(\text{children}, \text{parent}) = \begin{cases} 
\text{if} & \text{listp}(\text{children}) \\
\text{then} & \text{cons}(\text{cons}(\text{car}(\text{children}), \text{parent}), \text{ulnks}(\text{cdr}(\text{children}), \text{parent})) \\
\text{else} & \text{nil} \end{cases}
\]

**Theorem:** all-done-and-all-empty-implies-number-not-reported-0

\( (\text{all-done}(\text{children}, \text{state}) \land \text{all-empty}(\text{ulnks}(\text{children}, \text{parent}), \text{state})) \rightarrow (\text{number-not-reported}(\text{children}, \text{parent}, \text{state}) = 0) \)

**Theorem:** all-empty-implies-all-empty-sublist

\( (\text{all-empty}(\text{channels}, \text{state}) \land \text{sublistp}(\text{sublist}, \text{channels})) \rightarrow \text{all-empty}(\text{sublist}, \text{state}) \)

**Theorem:** sublist-ulnks

\[ \begin{align*}
\text{proper-tree}(\text{tree}, \text{tree}) & \land \text{sublistp}(\text{sublist}, \text{children}(\text{parent}, \text{tree})) \\
& \land \text{setp}(\text{nodes}(\text{tree})) \\
\rightarrow & \text{sublistp}(\text{ulnks}(\text{sublist}, \text{parent}), \text{up-links}(\text{cdr}(\text{nodes}(\text{tree})), \text{tree}))
\end{align*} \]

**Theorem:** child-of-node-in-suffix-is-in-suffix

\[ \begin{align*}
\text{proper-tree}(\text{tree}, \text{tree}) & \land \text{setp}(\text{nodes}(\text{tree})) \\
& \land (\text{child} \in \text{children}(\text{node}, \text{tree})) \\
& \land \text{suffix}(\text{nodes}, \text{nodes}(\text{tree})) \\
& \land (\text{node} \in \text{nodes}) \\
\rightarrow & (\text{child} \in \text{cdr}(\text{nodes}))
\end{align*} \]
Theorem: children-are-suffix-of-sublist-generalized
(proper-tree ('tree, tree)
∧ setp (nodes (tree))
∧ suffix (nodes, nodes (tree))
∧ (node ∈ nodes)
∧ sublistp (sublist, children-rec ('tree, node, tree))
→ sublistp (sublist, cdr (nodes))

Theorem: all-nodes-are-done
(proper-tree ('tree, tree)
∧ setp (nodes (tree))
∧ all-empty (down-links (nodes (tree), tree, state)
∧ all-empty (up-links (cdr (nodes (tree)), tree, state)
∧ dl (down-links (nodes (tree), tree, state)
∧ ul (up-links (cdr (nodes (tree)), tree), state)
∧ no (nodes (tree), tree, state)
∧ (status (car (tree), state) = 'started)
∧ suffix (nodes, nodes (tree)))
→ all-done (nodes, state)

Theorem: all-done-implies-total-outstanding-0
all-done (nodes, state) → (total-outstanding (nodes, tree, state) = 0)

Theorem: all-empty-root-started-implies-total-outstanding-0
(proper-tree ('tree, tree)
∧ setp (nodes (tree))
∧ inv (tree, state)
∧ all-empty (down-links (nodes (tree), tree, state)
∧ all-empty (up-links (cdr (nodes (tree)), tree, state)
∧ (status (car (tree), state) = 'started))
→ (total-outstanding (nodes (tree), tree, state) = 0)

Definition:
full-channel (channels, state)
= if listp (channels)
    then if empty (car (channels), state)
        then full-channel (cdr (channels), state)
        else car (channels) endif
    else f endif

Theorem: not-all-empty-implies-full-channel-full
((¬ all-empty (channels, state)) ∧ (f ∉ channels))
→ (listp (cdr (assoc (full-channel (channels, state), state)))
∧ (full-channel (channels, state) ∈ channels)
∧ full-channel (channels, state))
THEOREM: not-total-outstanding-0-implies-full-channel
(proper-tree ('tree, tree)
\land setp (nodes (tree))
\land inv (tree, state)
\land ((status (car (tree), state) = 'started)
\lor (status (car (tree), state) = 'not-started))
\land (total-outstanding (nodes (tree), tree, state) \neq 0))
\rightarrow ((status (car (tree), state) = 'not-started)
\lor full-channel (down-links (nodes (tree), tree, state))
\lor full-channel (up-links (cdr (nodes (tree)), tree, state)))

THEOREM: status-root-becomes-started-or-unchanged
(treep (tree) \land (statement \in tree-prg (tree)) \land n (old, new, statement))
\rightarrow ((status (car (tree), new) = 'started)
\lor (status (car (tree), new) = status (car (tree), old)))

THEOREM: root-started-or-not-started-is-invariant
(initial-condition ('(and
(all-empty ,(all-channels tree) state)
(not-started ,(nodes tree) state)),
tree-prg (tree))
\land treep (tree))
\rightarrow invariant ('(or
(equal
(status ,(car tree) state)
'started)
(equal
(status ,(car tree) state)
'not-started)),
tree-prg (tree))

THEOREM: total-outstanding-decreases-sublist
(treep (tree)
\land dl (down-links (nodes (tree), tree), old)
\land n (old, new, statement)
\land (statement \in tree-prg (tree))
\land sublistp (nodes, nodes (tree))
\land (node \in nodes)
\land (if status (node, new) = 'started then outstanding (node, new)
\hspace{1em} else 1 + length (children (node, tree)) endif
\hspace{1em} <
\hspace{1em} if status (node, old) = 'started
\hspace{1em} then outstanding (node, old)
\hspace{1em} else 1 + length (children (node, tree)) endif)
\rightarrow (total-outstanding (nodes, tree, new)
\hspace{1em} <
\hspace{1em} total-outstanding (nodes, tree, old))

45
**Definition:**

tou (old, new, node, tree) =
( if status (node, new) = 'started then outstanding (node, new)
else 1 + length (children (node, tree)) endif
<
if status (node, old) = 'started
then outstanding (node, old)
else 1 + length (children (node, tree)) endif)

**Theorem:** total-outstanding-decreases

treep (tree) ∧ dl (down-links (nodes (tree), tree), old) ∧ n (old, new, statement) ∧ (statement ∈ tree-prg (tree)) ∧ (node ∈ nodes (tree)) ∧ tou (old, new, node, tree)) → (total-outstanding (nodes (tree), tree, new) < total-outstanding (nodes (tree), tree, old))

**Theorem:** start-decreases-tou

treep (tree) ∧ (status (car (tree), old) = 'not-started) ∧ n (old, new, list ('start, car (tree), rfp (car (tree), children (car (tree), tree)))) → tou (old, new, car (tree), tree)

**Theorem:** others-preserve-root-not-started

treep (tree) ∧ n (old, new, statement) ∧ (statement ∈ tree-prg (tree)) ∧ (statement ≠ list ('start, car (tree), rfp (car (tree), children (car (tree), tree)))) → (cdr (assoc (cons ('status, car (tree)), new)) = status (car (tree), old))

**Theorem:** root-receive-report-decreases-tou

treep (tree) ∧ listp (channel (cons (child, car (tree)), old)) ∧ (child ∈ children (car (tree), tree)) ∧ n (old, new, list ('root-receive-report, car (tree), cons (child, car (tree)))) ∧ inv (tree, old)) → tou (old, new, car (tree), tree)
THEOREM: others-preserve-up-to-root-full
\[
(\text{treep}(\text{tree}) \\
\land \text{listp}(\text{channel}(\text{cons}(\text{child}, \text{car}(\text{tree})), \text{old}))) \\
\land (\text{child} \in \text{children}(\text{car}(\text{tree}), \text{tree})) \\
\land (\text{statement} \in \text{tree-prg}(\text{tree})) \\
\land (\text{statement} \neq \text{list}(\text{root-receive-report}, \text{car}(\text{tree}), \text{cons}(\text{child}, \text{car}(\text{tree})))) \\
\land \text{n}(\text{old}, \text{new}, \text{statement}) \\
\rightarrow \text{listp}\left(\text{cdr}\left(\text{assoc}\left(\text{cons}(\text{child}, \text{car}(\text{tree})), \text{new}\right)\right)\right)
\]

THEOREM: receive-find-decreases-tou
\[
(\text{treep}(\text{tree}) \\
\land \text{listp}(\text{channel}(\text{cons}(\text{parent}(\text{node}, \text{tree}), \text{node}), \text{old}))) \\
\land (\text{node} \in \text{cdr}(\text{nodes}(\text{tree}))) \\
\land \text{n}(\text{old}, \text{new}, \text{statement}) \\
\rightarrow \text{tou}(\text{old}, \text{new}, \text{node}, \text{tree})
\]

THEOREM: others-preserve-down-to-node-full
\[
(\text{treep}(\text{tree}) \\
\land \text{listp}(\text{channel}(\text{cons}(\text{parent}(\text{node}, \text{tree}), \text{node}), \text{old}))) \\
\land (\text{node} \in \text{cdr}(\text{nodes}(\text{tree}))) \\
\land (\text{statement} \in \text{tree-prg}(\text{tree})) \\
\land (\text{statement} \neq \text{list}(\text{receive-find}, \text{node}, \text{cons}(\text{node}, \text{parent}(\text{node}, \text{tree})), \text{rfp}(\text{node}, \text{children}(\text{node}, \text{tree})))) \\
\land \text{n}(\text{old}, \text{new}, \text{statement}) \\
\rightarrow \text{listp}(\text{cdr}(\text{assoc}(\text{cons}(\text{child}, \text{car}(\text{tree})), \text{new})))
\]

THEOREM: receive-report-decreases-tou
\[
(\text{treep}(\text{tree}) \\
\land \text{listp}(\text{channel}(\text{cons}(\text{child}, \text{node}), \text{old}))) \\
\land (\text{node} \in \text{cdr}(\text{nodes}(\text{tree}))) \\
\land (\text{child} \in \text{children}(\text{node}, \text{tree})) \\
\land \text{n}(\text{old}, \text{new}, \text{node}, \text{tree})
\]
list ('receive-report, 
node, 
cons (child, node), 
cons (node, parent (node, tree))))
∧ inv (tree, old))
→ tou (old, new, node, tree)

**Theorem:** others-preserve-up-to-node-full
(treep (tree)
∧ listp (channel (cons (child, node), old))
∧ (node ∈ cdr (nodes (tree)))
∧ (child ∈ children (node, tree))
∧ (statement ≠ list ('receive-report, 
node, 
cons (child, node), 
cons (node, parent (node, tree))))
∧ (statement ∈ tree-prg (tree))
∧ n (old, new, statement))
→ listp (cdr (assoc (cons (child, node), new)))

**Event:** Disable total-outstanding-decreases.

**Event:** Disable tou.

**Event:** Disable total-outstanding-decreases-sublist.

**Event:** Disable status-root-becomes-started-or-unchanged.

**Event:** Disable not-total-outstanding-0-implies-full-channel.

**Event:** Disable not-all-empty-implies-full-channel-full.

**Event:** Disable full-channel.

**Event:** Disable all-empty-root-started-implies-total-outstanding-0.

**Event:** Disable all-done-implies-total-outstanding-0.

**Event:** Disable all-nodes-are-done.


EVENT: Disable sublist-ulnks.

EVENT: Disable all-empty-implies-all-empty-sublist.

EVENT: Disable all-done-and-all-empty-implies-number-not-reported-0.

EVENT: Disable ulnks.

EVENT: Disable all-done-implies-all-done-sublist.

EVENT: Disable all-done-implies-done.

EVENT: Disable all-done.

EVENT: Disable later-positions-are-in-suffix.

EVENT: Disable setp-list-setp-suffix.

EVENT: Disable childs-position-increases.

EVENT: Disable member-suffix-member-list.

EVENT: Disable suffix-implies-suffix-cdr.

EVENT: Disable suffix.


EVENT: Disable parent-to-root-induction.

EVENT: Disable parents-position-decreases.
EVENT: Disable position-append.

EVENT: Disable total-outstanding-non-increasing.

EVENT: Disable total-outstanding-non-increasing-sublist.

EVENT: Disable outstanding-non-increasing.

EVENT: Disable dl-ul-no-preserves-no-sublist.

EVENT: Disable dl-ul-no-preserves-instance-of-no.

EVENT: Disable receive-report-preserves-instance-of-no.

EVENT: Disable child-member-cdr-nodes.

EVENT: Disable receive-report-preserves-no-for-parent.

EVENT: Disable receive-report-preserves-no-for-node.

EVENT: Disable receive-report-preserves-no-for-rest-of-tree.

EVENT: Disable receive-find-preserves-instance-of-no.

EVENT: Disable dl-down-links-implies-dl-rfp.

EVENT: Disable down-links-1-rfp.

EVENT: Disable dl-of-append.

EVENT: Disable receive-find-preserves-no-for-parent-of-node.

EVENT: Disable receive-find-preserves-no-for-node.

EVENT: Disable receive-find-preserves-no-for-rest-of-tree.

EVENT: Disable setp-nodes-setp-children.

EVENT: Disable setp-nodes-implies-setp-roots.

EVENT: Disable number-not-reported-of-root.

EVENT: Disable number-not-reported-of-non-root.

EVENT: Disable min-of-reported-of-non-root.

EVENT: Disable update-min-of-reported.

EVENT: Disable min-of-reported-of-min.

EVENT: Disable min-commutative-1.

EVENT: Disable min-associative.

EVENT: Disable min-commutative.

EVENT: Disable start-preserves-instance-of-no.

EVENT: Disable length-rfp.

EVENT: Disable start-preserves-no-for-rest-of-tree.

EVENT: Disable unchanged-preserves-no.

EVENT: Disable start-preserves-no-for-parent.


EVENT: Disable dl-ul-no-preserves-ul.
EVENT: Disable dl-ul-no-preserves-ul-sublist.


EVENT: Disable zero-not-reported-implies-children-reported.

EVENT: Disable member-up-links.

EVENT: Disable dl-preserves-dl.

EVENT: Disable dl-preserves-sublist.

EVENT: Disable dl-preserves-instance-of-dl.

EVENT: Disable all-numberps-nodes-implies-all-numberps-children.

EVENT: Disable all-numberps-forest-implies-all-numberps-roots.

EVENT: Disable parent-not-litatom.


EVENT: Disable all-numberps-nodes-implies-all-numberps-parent.

EVENT: Disable all-numberps-append.

EVENT: Disable send-find-general.

EVENT: Disable assoc-equal-cons.

EVENT: Disable send-find-implies.

EVENT: Disable member-rfp.

EVENT: Disable parent-not-grandchild.

EVENT: Disable parent-not-child.

EVENT: Disable member-down-links.

EVENT: Disable member-down-links-1.


EVENT: Disable no-implies-instance-of-no.

EVENT: Disable ul-implies-instance-of-ul.

EVENT: Disable dl-implies-instance-of-dl.

EVENT: Disable inv-implies-augmented-correctness-condition.

EVENT: Disable initial-conditions-imply-invariant.

EVENT: Disable all-empty-not-started-implies-dl.

EVENT: Disable inv.

EVENT: Disable dl.

EVENT: Disable node-values-constant-invariant.

EVENT: Disable node-values-constant-unless-sufficient.

EVENT: Disable listp-tree-prg.

EVENT: Disable root-receive-report-prg-is-total.

EVENT: Disable start-prg-is-total.
EVENT: Disable receive-report-prg-is-total.

EVENT: Disable receive-find-prg-is-total.


EVENT: Disable root-receive-report-func.

EVENT: Disable start-func-implements-start.

EVENT: Disable start-func.


EVENT: Disable receive-report-func.

EVENT: Disable receive-find-func-implements-receive-find.

EVENT: Disable uc-of-send-find-func.

EVENT: Disable to-node-not-in-rfp.

EVENT: Disable parent-not-in-rfp.

EVENT: Disable about-rfp-numberp.

EVENT: Disable about-rfp.

EVENT: Disable assoc-of-send-find-func.

EVENT: Disable send-find-of-update-assoc.

EVENT: Disable children-are-not-litatoms-member.

EVENT: Disable children-are-not-litatoms.
EVENT: Disable parent-is-not-a-litatom.

EVENT: Disable nodes-are-not-litatoms.

EVENT: Disable send-find-func-implements-send-find.

EVENT: Disable receive-find-func.

EVENT: Disable send-find-func.

EVENT: Disable no-at-termination.

EVENT: Disable found-value-min-value-generalized.

EVENT: Disable min-of-two-nodes-values.


EVENT: Disable number-not-reported-0-implies.

EVENT: Disable total-outstanding-0-implies.

EVENT: Disable no-implies.

EVENT: Disable found-value-node-value-append.

EVENT: Disable nati.

EVENT: Disable found-value-node-value.

EVENT: Disable down-links-is-sublistp.

EVENT: Disable children-of-non-node.

EVENT: Disable sublistp-down-links-1.
EVENT: Disable sublistp-not-started.


EVENT: Disable nodes-in-channels-append.


EVENT: Disable not-started-implies-no.

EVENT: Disable nodes-in-channels.

EVENT: Disable all-empty-implies-ul.

EVENT: Disable all-empty-append.

EVENT: Disable not-started-implies-not-started.

EVENT: Disable all-empty-implies-empty.

EVENT: Disable correct.

EVENT: Disable min-node-value.

EVENT: Disable all-empty.

EVENT: Disable all-channels.

EVENT: Disable not-started.

EVENT: Disable up-links.

EVENT: Disable down-links.

EVENT: Disable down-links-1.
EVENT: Disable no.

EVENT: Disable min-of-reported.

EVENT: Disable number-not-reported.

EVENT: Disable reported.

EVENT: Disable ul.

EVENT: Disable done.

EVENT: Disable total-outstanding.

EVENT: Disable treep.

EVENT: Disable member-tree-prg.

EVENT: Disable equal-if.

EVENT: Disable tree-prg.

EVENT: Disable member-root-receive-report-prg.


EVENT: Disable member-rrrp.

EVENT: Disable rrrp.

EVENT: Disable member-start-prg.

EVENT: Disable start-prg.

EVENT: Disable member-receive-report-prg.
EVENT: Disable receive-report-prg.

EVENT: Disable member-rrp.

EVENT: Disable rrp.

EVENT: Disable member-receive-find-prg.

EVENT: Disable receive-find-prg.

EVENT: Disable rfp.

EVENT: Disable root-receive-report.

EVENT: Disable start.

EVENT: Disable receive-report.

EVENT: Disable min.

EVENT: Disable receive-find.

EVENT: Disable send-find.

EVENT: Disable node-value.

EVENT: Disable outstanding.

EVENT: Disable found-value.

EVENT: Disable status.

EVENT: Disable receive.

EVENT: Disable send.
EVENT: Disable head.

EVENT: Disable empty.

EVENT: Disable channel.

EVENT: Disable value.


EVENT: Disable parent-is-not-child.

EVENT: Disable listp-parent-rec-equals.

EVENT: Disable parent-is-not-itself.

EVENT: Disable parent-is-not-itself-generalized.

EVENT: Disable node-has-parent.

EVENT: Disable children-of-setp-tree.

EVENT: Disable member-subtree-member-tree.


EVENT: Disable not-member-no-children.

EVENT: Disable not-member-subtrees.

EVENT: Disable proper-tree-next-level-of-proper-tree.

EVENT: Disable proper-tree-of-append.

EVENT: Disable next-level-in-subtrees-forest.


EVENT: Disable subtrees-of-subtree-in-complete-subtrees.


EVENT: Disable next-level-reduces-count.

EVENT: Disable nodes-rec-forest-append.

EVENT: Disable next-level.

EVENT: Disable subtreep-subtrees.

EVENT: Disable subtrees.

EVENT: Disable subtreep.

EVENT: Disable sublistp-children.

EVENT: Disable sublistp-children-generalized.

EVENT: Disable node-that-has-parent-is-in-tree.

EVENT: Disable node-that-has-child-is-in-tree.

EVENT: Disable member-parent-member-tree.

EVENT: Disable parent-of-child.

EVENT: Disable member-parent-parent.
EVENT: Disable member-child-tree.

EVENT: Disable not-member-no-parent.

EVENT: Disable plistp-roots.

EVENT: Disable plistp-parent-rec.

EVENT: Disable plistp-children-rec.

EVENT: Disable member-roots-member-forest.


EVENT: Disable not-flag-tree.

EVENT: Disable canonicalize-children-rec-flag.

EVENT: Disable canonicalize-parent-rec-flag.

EVENT: Disable canonicalize-proper-tree-flag.

EVENT: Disable canonicalize-nodes-rec-flag.

EVENT: Disable proper-tree.

EVENT: Disable parent.

EVENT: Disable parent-rec.

EVENT: Disable children.

EVENT: Disable children-rec.

EVENT: Disable roots.
EVENT: Disable nodes.

EVENT: Disable nodes-rec.

EVENT: Disable sublistp-in-cons.

EVENT: Disable sublistp-in-append.

EVENT: Disable sublistp-reflexive.

EVENT: Disable sublistp-easy.

EVENT: Disable sei.

EVENT: Disable sublistp-normalize.

EVENT: Disable sublistp-of-sublistp-is-sublistp.

EVENT: Disable member-of-sublistp-is-member.

EVENT: Disable sublistp-append.

EVENT: Disable sublistp.

EVENT: Disable setp-member-2.

EVENT: Disable setp-member-1.

EVENT: Disable setp-append-canonicalize.

EVENT: Disable setp-append-not-listp.

EVENT: Disable setp-append-cons.

EVENT: Disable setp-member.
EVENT: Disable setp-append.

EVENT: Disable setp.

EVENT: Disable all-numberps-implies.

EVENT: Disable all-numberps.

EVENT: Disable not-lessp-count-append.

EVENT: Disable append-plistp-nil.

EVENT: Disable plistp-append-plistp.

EVENT: Disable plistp.

EVENT: Disable length-append.

EVENT: Disable listp-append.

EVENT: Disable car-append.

EVENT: Disable n.

THEOREM: member-cdr-nodes-member-nodes
\[(\text{node} \in \text{cdr (nodes (tree))}) \land \text{treep (tree)} \rightarrow (\text{node} \in \text{nodes (tree)})\]

THEOREM: total-outstanding-decreases-expanded
\[(\text{treep (tree)}
\land \ \text{inv (tree, old)}
\land \ \text{n (old, new, statement)}
\land \ (\text{statement} \in \text{tree-prg (tree)})
\land \ (\text{node} \in \text{nodes (tree)})
\land \ \text{tou (old, new, node, tree)}
\rightarrow ((\text{total-outstanding (nodes (tree), tree, new)}
\quad < \quad \text{total-outstanding (nodes (tree), tree, old)})
\quad = \quad t)
\land \ (\text{total-outstanding (nodes (tree), tree, new)}\]
<  total-outstanding\(\text{nodes}\left(\text{tree},\ old\right))\)

**Theorem:** total-outstanding-decreases-expanded-count
\[
\text{treep}\left(\text{tree}\right) \land \text{inv}\left(\text{tree},\ old\right) \land \text{n}\left(\text{old},\ new,\ \text{statement}\right) \land \left(\text{statement} \in \text{tree-prg}\left(\text{tree}\right)\right) \land \left(\text{node} \in \text{nodes}\left(\text{tree}\right)\right) \land \text{tou}\left(\text{old},\ new,\ \text{node},\ \text{tree}\right) \land \left(\text{total-outstanding}\left(\text{nodes}\left(\text{tree},\ \text{tree},\ \text{old}\right)\right) = (1 + \text{count})\right) \\
\rightarrow \left(\left(\text{total-outstanding}\left(\text{nodes}\left(\text{tree},\ \text{tree},\ \text{new}\right)\right) < (1 + \text{count})\right) = \text{t}\right) \land \left(\text{total-outstanding}\left(\text{nodes}\left(\text{tree},\ \text{tree},\ \text{new}\right)\right) < (1 + \text{count})\right)
\]

**Theorem:** total-outstanding-non-increasing-expanded
\[
\text{treep}\left(\text{tree}\right) \land \left(\text{statement} \in \text{tree-prg}\left(\text{tree}\right)\right) \land \text{n}\left(\text{old},\ new,\ \text{statement}\right) \land \text{inv}\left(\text{tree},\ old\right) \\
\rightarrow \left(\left(\text{total-outstanding}\left(\text{nodes}\left(\text{tree},\ \text{tree},\ \text{old}\right)\right) < \text{total-outstanding}\left(\text{nodes}\left(\text{tree},\ \text{tree},\ \text{new}\right)\right)\right) = \text{f}\right) \land \left(\text{total-outstanding}\left(\text{nodes}\left(\text{tree},\ \text{tree},\ \text{old}\right)\right) \neq \text{total-outstanding}\left(\text{nodes}\left(\text{tree},\ \text{tree},\ \text{new}\right)\right)\right)
\]

**Theorem:** total-outstanding-non-increasing-expanded-count
\[
\text{treep}\left(\text{tree}\right) \land \left(\text{statement} \in \text{tree-prg}\left(\text{tree}\right)\right) \land \text{n}\left(\text{old},\ new,\ \text{statement}\right) \land \text{inv}\left(\text{tree},\ old\right) \land \left(\text{total-outstanding}\left(\text{nodes}\left(\text{tree},\ \text{tree},\ \text{old}\right)\right) = (1 + \text{count})\right) \\
\rightarrow \left(\left(1 + \text{count} < \text{total-outstanding}\left(\text{nodes}\left(\text{tree},\ \text{tree},\ \text{old}\right)\right)\right) = \text{f}\right) \land \left(\left(1 + \text{count}\right) \neq \text{total-outstanding}\left(\text{nodes}\left(\text{tree},\ \text{tree},\ \text{new}\right)\right)\right)
\]

**Theorem:** key-statements-member-tree-prg
\[
\text{treep}\left(\text{tree}\right) \land \left(\text{statement} \in \text{tree-prg}\left(\text{tree}\right)\right) \land \text{n}\left(\text{old},\ new,\ \text{statement}\right) \land \text{inv}\left(\text{tree},\ old\right) \land \left(\text{total-outstanding}\left(\text{nodes}\left(\text{tree},\ \text{tree},\ \text{old}\right)\right) = (1 + \text{count})\right) \\
\rightarrow \left(\left(\text{list}\left(\text{'start},\ \text{car}\left(\text{tree}\right),\ \text{rfp}\left(\text{car}\left(\text{tree}\right),\ \text{children}\left(\text{car}\left(\text{tree}\right),\ \text{tree}\right)\right)\right) \in \text{tree-prg}\left(\text{tree}\right)\right)\right) \land \left(\text{tree}\left(\text{tree}\right) \land \left(\text{child} \in \text{children}\left(\text{car}\left(\text{tree}\right),\ \text{tree}\right)\right)\right) \\
\rightarrow \left(\text{list}\left(\text{'root-receive-report},\ \text{car}\left(\text{tree}\right),\ \text{cons}\left(\text{child},\ \text{car}\left(\text{tree}\right)\right) \in \text{tree-prg}\left(\text{tree}\right)\right)\right) \land \left(\text{tree}\left(\text{tree}\right) \land \left(\text{node} \in \text{cdr}\left(\text{nodes}\left(\text{tree}\right)\right)\right)\right) \\
\rightarrow \left(\text{list}\left(\text{'receive-find},\ \text{node}\right)\right)
\]
cons (parent (node, tree), node),
cons (node, parent (node, tree)),
rfp (node, children (node, tree))
∈ tree-prg (tree))
∧ (treep (tree)
∧ (node ∈ cdr (nodes (tree)))
∧ (child ∈ children (node, tree)))
→ (list (receive-report, node,
cons (child, node),
cons (node, parent (node, tree)))
∈ tree-prg (tree))

THEOREM: down-link-full-decreases-total-outstanding-ensures (treep (tree)
∧ (node ∈ cdr (nodes (tree)))
∧ inv (tree, old)
∧ listp (channel (cons (parent (node, tree), node), old))
∧ n (old, new,
list (receive-find, node,
cons (parent (node, tree), node),
cons (node, parent (node, tree)),
rfp (node, children (node, tree))))
→ (total-outstanding (nodes (tree), tree, new) <
total-outstanding (nodes (tree), tree, old))

THEOREM: down-link-full-unless (treep (tree)
∧ (node ∈ cdr (nodes (tree)))
∧ listp (channel (cons (parent (node, tree), node), old))
∧ (statement ∈ tree-prg (tree))
∧ (statement ≠ list (receive-find, node,
cons (parent (node, tree), node),
cons (node, parent (node, tree)),
rfp (node, children (node, tree))))
∧ n (old, new, statement))
→ listp (channel (cons (parent (node, tree), node), new))

THEOREM: down-link-full-decreases-total-outstanding (treep (tree) ∧ (node ∈ cdr (nodes (tree))))
→ leads-to (and (inv tree state)
(and
  (listp
   (channel
    ',(cons (parent node tree) node)
     state))
  (equal
   (total-outstanding
    ',(nodes tree)
     tree
    state)
   ',(add1 count))
  (lessp
   (total-outstanding
    ',(nodes tree)
     tree
    state)
   ',(add1 count))
  tree-prg (tree))

THEOREM: member-car-tree-nodes-tree
  treep (tree) → (car (tree) ∈ nodes (tree))

THEOREM: root-up-link-full-decreases-total-outstanding-ensures
  (treep (tree)
    ∧ inv (tree, old)
    ∧ (child ∈ children (car (tree), tree))
    ∧ listp (channel (cons (child, car (tree)), old))
    ∧ n (old, new, list ('root-receive-report, car (tree), cons (child, car (tree))))
    → (total-outstanding (nodes (tree), tree, new)
    < total-outstanding (nodes (tree), tree, old))

THEOREM: root-up-link-full-unless
  (treep (tree)
    ∧ (child ∈ children (car (tree), tree))
    ∧ listp (channel (cons (child, car (tree)), old))
    ∧ (statement ∈ tree-prg (tree))
    ∧ (statement ≠ list ('root-receive-report, car (tree), cons (child, car (tree))))
    ∧ n (old, new, statement)
    → listp (channel (cons (child, car (tree)), new))

THEOREM: up-link-full-decreases-total-outstanding-ensures
(treep (tree)
  ∧ inv (tree, old)
  ∧ (node ∈ cdr (nodes (tree)))
  ∧ (child ∈ children (node, tree))
  ∧ listp (channel (cons (child, node), old))
  ∧ n (old, new, list ('receive-report, node, cons (child, node), cons (node, parent (node, tree))))
  → (total-outstanding (nodes (tree), tree, new) < total-outstanding (nodes (tree), tree, old))

THEOREM: up-link-full-unless
(treep (tree)
  ∧ (node ∈ cdr (nodes (tree)))
  ∧ (child ∈ children (node, tree))
  ∧ listp (channel (cons (child, node), old))
  ∧ (statement ∈ tree-prg (tree))
  ∧ (statement ≠ list ('receive-report, node, cons (child, node), cons (node, parent (node, tree))))
  ∧ n (old, new, statement))
  → listp (channel (cons (child, node), new))

THEOREM: member-cdr-nodes-equals
(treep (tree) ∧ (node ≠ car (tree)))
  → ((node ∈ nodes (tree)) = (node ∈ cdr (nodes (tree))))

THEOREM: up-link-full-decreases-total-outstanding
(treep (tree) ∧ (node ∈ nodes (tree)) ∧ (child ∈ children (node, tree)))
  → leads-to ('(and
    (inv ',tree state)
  (and
    (listp
      (channel ',(cons child node) state))
    (equal
      (total-outstanding ',(nodes tree) '
        ,tree state)
      ',(add1 count)))},
    'lessenp

67
\begin{align*}
&\text{(total-outstanding, (nodes tree), tree state), (add1 count)), tree-prg (tree)} \\
\text{Theorem: not-started-root-decreases-total-outstanding-ensures} \\
&\text{treep (tree)} \\
&\land \text{ inv (tree, old)} \\
&\land \text{ (status (car (tree), old) = 'not-started)} \\
&\land \text{ n (old, new,} \\
&\text{ list ('start, car (tree), rfp (car (tree), children (car (tree), tree)))))} \\
&\rightarrow \text{ (total-outstanding (nodes (tree), tree, new) < total-outstanding (nodes (tree), tree, old))} \\
\text{Theorem: not-started-root-unless} \\
&\text{treep (tree)} \\
&\land \text{ (status (car (tree), old) = 'not-started)} \\
&\land \text{ (statement } \in \text{ tree-prg (tree))} \\
&\land \text{ (statement } \neq \text{ list ('start, car (tree), rfp (car (tree), children (car (tree), tree)))))} \\
&\land \text{ n (old, new, statement))} \\
&\rightarrow \text{ (status (car (tree), new) = 'not-started)} \\
\text{Theorem: not-started-root-decreases-total-outstanding} \\
&\text{treep (tree)} \\
&\rightarrow \text{ leads-to ('(and' \\
&\text{ (inv ',tree state))} \\
&\text{ (and} \\
&\text{ (equal} \\
&\text{ (status ',(car tree) state) 'not-started}) \\
&\text{ (equal} \\
&\text{ (total-outstanding, (nodes tree) tree state, (add1 count))}}, \\
&\text{'(lessp} \\
&\text{ (total-outstanding, (nodes tree) tree)} \\
&\text{68}
\end{align*}
state)
,',(add1 count)),
tree-prg(tree))

THEOREM: full-channel-not-f-implies
full-channel(channels, state)
→ ((full-channel(channels, state) ∈ channels)
   ∧ listp(channel(full-channel(channels, state), state)))

THEOREM: total-outstanding-decreases-leads-to
(tree(tree) ∧ initial-condition(‘(and
   (all-empty
   ’,(all-channels tree)
   state)
   (not-started ’,(nodes tree) state)),
   tree-prg(tree))
→ leads-to(‘(equal
   (total-outstanding
   ’,(nodes tree)
   ’,tree
   state)
   ’,(add1 count)),
   ‘(lessp
   (total-outstanding
   ’,(nodes tree)
   ’,tree
   state)
   ’,(add1 count)),
   tree-prg(tree))

THEOREM: termination-induction
(tree(tree) ∧ initial-condition(‘(and
   (all-empty
   ’,(all-channels tree)
   state)
   (not-started ’,(nodes tree) state)),
   tree-prg(tree))
→ leads-to(‘(lessp
   (total-outstanding
   ’,(nodes tree)
   ’,tree
   state)
   ’,(add1 count)),
   tree-prg(tree))
Theorem: termination
(tree (tree))
∧ initial-condition ('(and
    (all-empty
    ',(all-channels tree)
    state)
    (not-started ',(nodes tree) state)),
    tree-prg (tree))
→ leads-to ('(true),
  '('
    (total-outstanding
    ',(nodes tree)
    ',tree
    state)
    0),
    tree-prg (tree))

Theorem: correctness-condition
(tree (tree))
∧ initial-condition ('(and
    (all-empty
    ',(all-channels tree)
    state)
    (not-started ',(nodes tree) state)),
    tree-prg (tree))
→ leads-to ('(true), '('
    (correct ',tree state),
    tree-prg (tree))

70
Index

about-rfp, 23
about-rfp-numberp, 23
all-channels, 18, 20
all-done, 43, 44
all-done-and-all-empty-implies-number-not-reported-0, 43
all-done-implies-all-done-sublist, 43
all-done-implies-done, 43
all-done-implies-total-outstanding-0, 44
all-empty, 18–20, 31, 42–44
all-empty-append, 19
all-empty-implies-all-empty-sublist, 43
all-empty-implies-empty, 19
all-empty-implies-ul, 19
all-empty-not-started-implies-dl, 19
all-empty-root-started-implies-total-outstanding-0, 44
all-nodes-are-done, 44
all-numberps, 1, 16, 23, 29, 30, 33, 34, 38
all-numberps-append, 29
all-numberps-do-not-contain-litatom, 29
all-numberps-forest-implies-all-numberps-roots, 29
all-numberps-implies, 1
all-numberps-nodes-implies-all-numberps-car-parent, 29
numberps-children, 30
numberps-parent, 29
append-plistp-nil, 1
assoc-equal-cons, 29
assoc-of-channel-preserved-root-receive-report, 29
assoc-of-send-find-func, 23
canonicalize-children-rec-flag, 4
canonicalize-nodes-rec-flag, 4
canonicalize-parent-rec-flag, 4
canonicalize-proper-tree-flag, 4
car-append, 1
changed, 10–12, 32–39
channel, 9–12, 16, 19, 27, 30, 31, 33, 34, 38, 46–48, 65–67, 69
child-member-cdr-nodes, 39
child-of-node-in-suffix-is-in-suffix, 43
children, 3, 12–17, 21, 28, 32, 33, 35, 37–41, 43, 45–48, 64–68
children-are-not-litatoms, 23
children-are-not-litatoms-members, 23
children-are-suffix-of-sublist-generalized, 44
children-of-non-node, 20
children-of-setp-tree, 8
children-rec, 3–6, 8, 9, 20, 23, 28, 30, 35, 39, 42, 44
childs-position-increases, 42
correct, 18, 22
correctness-condition, 70
dl, 16, 18, 19, 27, 30–32, 37, 39–42, 44–46
dl-and-all-empty-implies-root-defines-status, 42
dl-down-links-implies-dl-rfp, 37
dl-implies-instance-of-dl, 27
dl-of-append, 37
dl-preserves-dl, 30
dl-preserves-instance-of-dl, 30
dl-preserves-sublist, 30
dl-ul-no-preserves-instance-of-no, 40
ul, 30
dl-ul-no-preserves-no-sublist, 40
dl-ul-no-preserves-ul, 31
dl-ul-no-preserves-ul-sublist, 31
done, 16, 27, 31, 33, 34, 38, 43
down-link-full-decreases-total-
outstanding, 65
  outstanding-ensures, 65
down-link-full-unless, 65
down-links, 17–20, 28, 30, 31, 37,
  39–42, 44–46
down-links-1, 17, 19, 20, 28, 37
down-links-1-rfp, 37
down-links-is-sublistp, 20
empty, 9, 11, 16, 18, 24, 25, 27, 30,
  31, 36, 39, 44
equal-if, 15
found-value, 10–12, 16–18, 20, 24,
  25, 27, 31–35, 37–40
found-value-min-value-generalize
d, 21
found-value-node-value, 20–22
found-value-node-value-append, 20
full-channel, 44, 45, 69
full-channel-not-f-implies, 69
head, 9–12, 22, 24, 25
initial-condition, 27, 40, 45, 69, 70
initial-conditions-implies-invari
ant, 20
inv, 18, 20, 22, 40, 44–48, 63–68
inv-implies-augmented-correctne
ss-condition, 22
inv-is-invariant, 40
inv-preserve-inv, 40
invariant, 27, 41, 45
key-statements-member-tree-prg, 64
later-positions-are-in-suffix, 42
leads-to, 66, 68–70
length, 1, 10, 11, 16, 22, 25, 32, 36,
  41, 45, 46
length-append, 1
length-rfp, 32
listp-append, 1
listp-parent-rec-equals, 9
listp-tree-prg, 26
member-car-tree-nodes-tree, 66
member-cdr-tree-nodes-equals, 67
member-cdr-tree-nodes-member-nodes, 63
member-child-tree, 5
member-down-links, 28
member-down-links-1, 28
member-of-sublistp-is-member, 2
member-parent-member-tree, 6
member-parent-parent, 6
member-receive-find-prg, 12
member-receive-report-prg, 13
member-rfp, 28
member-root-receive-report-prg, 14
member-roots-member-forest, 5
member-rrp, 13
member-rrrp, 14
member-subtree-member-tree, 8
member-suffix-member-list, 42
member-tree-prg, 15
member-up-links, 30
min, 11, 12, 17, 18, 21, 24, 25, 33,
  39
min-associative, 33
min-commutative, 33
min-commutative-1, 33
min-node-value, 18, 20, 21
min-of-reported, 17, 21, 28, 32–40
min-of-reported-of-min, 33
min-of-reported-of-non-root, 33
min-of-two-nodes-values, 21
n, 24, 25, 30, 31, 40, 41, 45–48, 63–
68
nati, 20
next-level, 7, 8, 20
next-level-in-subtrees-forest, 7
next-level-of-subtrees-in-complete-
subtrees, 7
next-level-of-tree-in-subtrees, 7
next-level-reduces-count, 7
no, 17–19, 21, 22, 28, 31, 39, 40, 44
no-at-termination, 21
no-children-in-rest-of-forest, 8
no-children-in-rest-of-tree, 8
no-implies, 21
no-implies-instance-of-no, 28
node-has-parent, 8
node-that-has-child-is-in-tree, 6
node-that-has-parent-is-in-tree, 6
node-value, 10, 11, 17, 18, 20–22, 25, 28, 33, 35, 37, 38, 40
node-values-constant-invariant, 27
node-values-constant-unless-sufficient, 26
nodes, 3, 14–16, 18–20, 22, 24, 26, 27, 30–32, 35, 37, 39–48, 63–68
nodes-are-not-litatoms, 23
nodes-in-channels, 19, 20
nodes-in-channels-append, 19
nodes-in-down-links-1-in-nodes, 19
nodes-in-down-links-in-nodes, 19
nodes-rec, 3–9, 20–23, 28–30, 35, 39, 41, 42
nodes-rec-forest-append, 7
not-all-empty-implies-full-channel-full, 44
not-flag-tree, 5
not-lessp-count-append, 1
not-member-no-children, 8
not-member-no-parent, 5
not-member-subtrees, 8
not-started, 18–20, 31, 36
not-started-implies-no, 19
not-started-implies-not-started, 19
not-started-root-decreases-total-outstanding, 68
not-started-root-decreases-total-outstanding-ensures, 68
not-started-root-unless, 68
not-total-outstanding-0-implies-full-channel, 45
number-not-reported, 17, 21, 28, 30, 32–40, 43
number-not-reported-0-implies, 21
number-not-reported-of-non-root, 34
number-not-reported-of-root, 34
others-preserve-down-to-node-full, 47
others-preserve-root-not-started, 46
others-preserve-up-to-node-full, 48
others-preserve-up-to-root-full, 47
outstanding, 10–12, 16, 17, 24, 25, 30, 32, 33, 35–41, 45, 46
outstanding-non-increasing, 41
parent, 4, 12, 13, 15, 18, 30, 37, 39, 42, 47, 48, 65, 67
parent-is-not-a-litatom, 23
parent-is-not-child, 9
parent-is-not-itself, 9
parent-is-not-itself-generalized, 8
parent-not-child, 28
parent-not-grandchild, 28
parent-not-in-children, 9
parent-not-in-rfp, 23
parent-not-litatom, 29
parent-not-started-implies-all-empty-and-not-started, 31
parent-of-child, 6
parent-of-parent-not-node, 28
parent-rec, 4–6, 8, 9, 23, 28, 29, 41, 47
parent-rec-children-rec, 5
parent-to-root-induction, 42
parents-position-decreases, 41
plistp, 1, 2, 5
plistp-append-plistp, 1
plistp-children-rec, 5
plistp-parent-rec, 5
plistp-roots, 5
position, 41–43
position-append, 41
proper-tree, 4–9, 16, 19–23, 28, 35, 39, 41–45
proper-tree-next-level-of-proper
r-tree, 8
proper-tree-of-append, 8
proper-tree-tree-implies-nodes-exists, 21
receive, 9–12, 22, 24, 25, 33, 34, 38
receive-find, 10, 37
receive-find-decreases-tou, 47
receive-find-func, 22, 24, 26
receive-find-func-implements-receive-find, 24
receive-find-preserves-instance-of-no, 37
receive-find-preserves-no-for-node, 36
receive-find-preserves-no-for-parent, 36
receive-find-preserves-no-for-rest-of-tree, 35
receive-find-prg, 12, 14, 24, 26
receive-report, 11, 39
receive-report-decreases-tou, 47
receive-report-func, 24–26
receive-report-func-implements-receive-report, 24
receive-report-preserves-instance-of-no, 39
receive-report-preserves-no-for-node, 38
receive-report-preserves-no-for-parent, 39
receive-report-preserves-no-for-rest-of-tree, 38
receive-report-prg, 13, 14, 24, 26
receive-report-prg-is-total, 26
reported, 16, 17, 21
rfp, 12–15, 23, 24, 28, 31, 32, 36, 37, 46, 47, 64, 65, 68
root-receive-report, 11, 35
root-receive-report-decreases-tou, 46
root-receive-report-func, 25, 26
root-receive-report-preserves-instance-of-no, 35
root-receive-report-prg, 14, 25, 26
root-receive-report-prg-is-total, 1, 26
root-started-or-not-started-is-invariant, 45
root-up-link-full-decreases-total-outstanding-ensures, 66
root-up-link-full-unless, 66
roots, 3–5, 8, 9, 21, 29, 35, 41
rrp, 13
rrrp, 14
sei, 2
send, 9–11, 22, 24, 29
send-find, 10, 11, 23, 29
send-find-func, 22–25
send-find-func-implements-send-find, 23
send-find-general, 29
send-find-implies, 29
send-find-of-update-assoc, 23
setp, 1, 2, 5, 6, 8, 9, 16, 22, 23, 28, 29, 33–35, 38, 39, 41–45
setp-append, 2
setp-append-canonicalize, 2
setp-append-cons, 2
setp-append-not-listp, 2
setp-list-setp-suffix, 42
setp-member, 2
setp-member-1, 2
setp-member-2, 2
setp-nodes-implies-setp-roots, 35
setp-nodes-setp-children, 35
setp-tree-unique-parent, 5
start, 11, 32
start-decreases-tou, 46
start-func, 25, 26
start-func-implements-start, 25
start-preserves-instance-of-no, 32
start-preserves-no-for-parent, 31
start-preserves-no-for-rest-of-tree, 32
start-prg, 13, 14, 25, 26
start-prg-is-total, 26
status, 10, 11, 16–18, 21, 25, 27, 28, 30–32, 35–37, 39–42, 44–46, 68
status-root-becomes-started-or-unchanged, 45
sublist-ulinks, 43
sublistp, 2, 3, 6, 7, 20, 22, 24, 30, 31, 36, 40, 41, 43–45
sublistp-append, 2
sublistp-children, 6
sublistp-children-generalized, 6
sublistp-down-links-1, 20
sublistp-easy, 3
sublistp-in-append, 3
sublistp-in-cons, 3
sublistp-normalize, 2
sublistp-not-started, 20
sublistp-of-sublistp-is-sublistp, 2
sublistp-reflexive, 3
subtreep, 6, 7
subtreep-subtrees, 7
subtrees, 7, 8, 22
subtrees-of-subtree-in-complete
  -subtrees, 7
subtrees-of-subtrees-in-complete
  -subtrees, 7
suffix, 42–44
suffix-implies-suffix-cdr, 42
termination, 70
termination-induction, 69
to-node-not-in-rfp, 24
total, 26
total-outstanding, 16, 21, 22, 41, 44–46, 63–68
total-outstanding-0-implies, 21
total-outstanding-decreases, 46
total-outstanding-decreases-expanded, 63
  -ended-count, 64
total-outstanding-decreases-lea
ds-to, 69
total-outstanding-decreases-sub

list, 45
total-outstanding-non-increasing
  g, 41
  g-expanded, 64
  g-expanded-count, 64
  g-sublist, 41
total-sufficient, 26
total-tree-prg, 26
tou, 46–48, 63, 64
tree-prg, 14, 15, 26, 27, 30, 31, 40, 41, 45–48, 63–70
treep, 16, 24–27, 30–32, 35, 37, 39–41, 45–48, 63–70
uc, 24, 29
uc-of-send-find-func, 24
ul, 16, 18, 19, 27, 31, 35, 39, 40, 44
ul-implies-instance-of-ul, 27
ul-implies-instance-of-ul-not-empty-uplink, 27
ulinks, 43
unchanged-preserves-no, 32
unless-sufficient, 27
up-link-full-decreases-total-outstanding, 67
  standing, 67
  standing-ensures, 67
up-link-full-unless, 67
up-links, 18, 30, 31, 35, 39, 40, 43–45
update-assoc, 22–25
update-min-of-reported, 33
value, 9, 10

zero-not-reported-implies-child
ren-reported, 30