EVENT: Start with the library `"interpreter"`.

```
;;;List Operations
```

THEOREM: car-append
$\text{car}(\text{append}(a,\ b))$
$= \ \textbf{if}\ \text{listp}(a)\ \textbf{then}\ \text{car}(a)$
$\quad\ \textbf{else}\ \text{car}(b)\ \textbf{endif}$

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:
$\text{plistp}(\mathit{list})$
$= \ \textbf{if}\ \text{listp}(\mathit{list})\ \textbf{then}\ \text{plistp}(\text{cdr}(\mathit{list}))$
$\quad\ \textbf{else}\ \mathit{list} = \textbf{nil}\ \textbf{endif}$

THEOREM: plistp-append-plistp
$\text{plistp}(\text{append}(a,\ b)) = \text{plistp}(b)$

THEOREM: append-plistp-nil
$(\text{append}(a,\ \textbf{nil}) = a) = \text{plistp}(a)$

THEOREM: not-lessp-count-append
$(\text{count}(x) + \text{count}(y)) \not< \text{count}(\text{append}(x,\ y))$

DEFINITION:
$\text{all-numberps}(\mathit{list})$
$= \ \textbf{if}\ \text{listp}(\mathit{list})$
$\quad\ \textbf{then}\ \textbf{if}\ \text{car}(\mathit{list}) \in \mathbf{N}\ \textbf{then}\ \text{all-numberps}(\text{cdr}(\mathit{list}))$
$\quad\quad\quad \textbf{else}\ \textbf{f}\ \textbf{endif}$
$\quad\ \textbf{else}\ \textbf{t}\ \textbf{endif}$

THEOREM: all-numberps-implies
$(\text{all-numberps}(\mathit{list}) \land (e \in \mathit{list})) \rightarrow (e \in \mathbf{N})$

```
;;;Set Operations
```

DEFINITION:
$\text{setp}(\mathit{list})$
$= \ \textbf{if}\ \text{listp}(\mathit{list})$
$\quad\ \textbf{then}\ \textbf{if}\ \text{car}(\mathit{list}) \in \text{cdr}(\mathit{list})\ \textbf{then}\ \textbf{f}$
$\quad\quad\quad \textbf{else}\ \text{setp}(\text{cdr}(\mathit{list}))\ \textbf{endif}$
$\quad\ \textbf{else}\ \textbf{t}\ \textbf{endif}$

THEOREM: setp-append
$((\neg\ \text{setp}\,(a)) \lor (\neg\ \text{setp}\,(b))) \rightarrow (\neg\ \text{setp}\,(\text{append}\,(a,\ b)))$

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
$(\text{setp}\,(\text{append}\,(a,\ b)) \land (x \in b)) \rightarrow (x \notin a)$

THEOREM: setp-member-2
$(\text{setp}\,(\text{append}\,(a,\ b)) \land (x \in a)) \rightarrow (x \notin b)$

`;;;Subset Operations`

DEFINITION:
$\text{sublistp}\,(\mathit{sub},\ \mathit{list})$
$= \ \ \textbf{if}\ \text{listp}\,(\mathit{sub})\ \ \textbf{then}\ (\text{car}\,(\mathit{sub}) \in \mathit{list}) \land \text{sublistp}\,(\text{cdr}\,(\mathit{sub}),\ \mathit{list})$
$\quad\ \textbf{else t endif}$

THEOREM: sublistp-append
$\text{sublistp}\,(\text{append}\,(a,\ b),\ \mathit{list}) = (\text{sublistp}\,(a,\ \mathit{list}) \land \text{sublistp}\,(b,\ \mathit{list}))$

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

DEFINITION:
$\text{sei}\,(a,\ b)$
$= \ \ \textbf{if}\ \text{listp}\,(a)\ \ \textbf{then}\ \text{sei}\,(\text{cdr}\,(a),\ \text{append}\,(b,\ \text{list}\,(\text{car}\,(a))))$
$\quad\ \textbf{else t endif}$

THEOREM: sublistp-easy
sublistp $(a,\ \text{append}\,(b,\ a))$

THEOREM: sublistp-reflexive
sublistp $(a,\ a)$

THEOREM: sublistp-in-append
$(\text{sublistp}\,(x,\ a) \lor \text{sublistp}\,(x,\ b)) \to \text{sublistp}\,(x,\ \text{append}\,(a,\ b))$

THEOREM: sublistp-in-cons
$\text{sublistp}\,(a,\ y) \to \text{sublistp}\,(a,\ \text{cons}\,(x,\ y))$

```
;;;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*))) **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 append (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* $\in$ roots (cdr (*tree*))
then cons (car (*tree*), parent-rec (`'forest`, *node*, cdr (*tree*)))
else parent-rec (`'forest`, *node*, cdr (*tree*)) **endif**
else append (parent-rec (`'tree`, *node*, car (*tree*)),
parent-rec (`'forest`, *node*, cdr (*tree*))) **endif**
else nil 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*))
$\wedge$ 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

$((\mathit{flag} \neq \texttt{'tree}) \land (\mathit{flag} \neq \texttt{'forest}))$
$\rightarrow \quad ((\text{nodes-rec}\,(\mathit{flag},\ \mathit{tree}) = \text{nodes-rec}\,(\texttt{'forest},\ \mathit{tree}))$
$\quad \land \quad (\text{proper-tree}\,(\mathit{flag},\ \mathit{tree}) = \text{proper-tree}\,(\texttt{'forest},\ \mathit{tree}))$
$\quad \land \quad (\text{parent-rec}\,(\mathit{flag},\ \mathit{child},\ \mathit{tree})$
$\quad\quad = \quad \text{parent-rec}\,(\texttt{'forest},\ \mathit{child},\ \mathit{tree}))$
$\quad \land \quad (\text{children-rec}\,(\mathit{flag},\ \mathit{parent},\ \mathit{tree})$
$\quad\quad = \quad \text{children-rec}\,(\texttt{'forest},\ \mathit{parent},\ \mathit{tree})))$

THEOREM: parent-rec-children-rec
$(\mathit{child} \in \text{children-rec}\,(\mathit{flag},\ \mathit{parent},\ \mathit{tree}))$
$= \quad (\mathit{parent} \in \text{parent-rec}\,(\mathit{flag},\ \mathit{child},\ \mathit{tree}))$

EVENT: Disable parent-rec-children-rec.

THEOREM: plistp-children-rec
$\text{plistp}\,(\text{children-rec}\,(\mathit{flag},\ \mathit{parent},\ \mathit{tree}))$

THEOREM: plistp-parent-rec
$\text{plistp}\,(\text{parent-rec}\,(\mathit{flag},\ \mathit{child},\ \mathit{tree}))$

THEOREM: plistp-roots
$\text{proper-tree}\,(\texttt{'forest},\ \mathit{forest}) \rightarrow \text{plistp}\,(\text{roots}\,(\mathit{forest}))$

THEOREM: member-roots-member-forest
$(\text{proper-tree}\,(\texttt{'forest},\ \mathit{forest}) \land (\mathit{node} \in \text{roots}\,(\mathit{forest})))$
$\rightarrow \quad (\mathit{node} \in \text{nodes-rec}\,(\texttt{'forest},\ \mathit{forest}))$

THEOREM: not-member-no-parent
$(\text{proper-tree}\,(\mathit{flag},\ \mathit{tree}) \land (\mathit{node} \notin \text{nodes-rec}\,(\mathit{flag},\ \mathit{tree})))$
$\rightarrow \quad (\text{parent-rec}\,(\mathit{flag},\ \mathit{node},\ \mathit{tree}) = \mathbf{nil})$

THEOREM: member-child-tree
$(\text{proper-tree}\,(\mathit{flag},\ \mathit{tree}) \land (\mathit{child} \in \text{children-rec}\,(\mathit{flag},\ \mathit{node},\ \mathit{tree})))$
$\rightarrow \quad (\mathit{child} \in \text{nodes-rec}\,(\mathit{flag},\ \mathit{tree}))$

THEOREM: setp-tree-unique-parent
$(\text{proper-tree}\,(\mathit{flag},\ \mathit{tree}) \land \text{setp}\,(\text{nodes-rec}\,(\mathit{flag},\ \mathit{tree})))$
$\rightarrow \quad (\text{parent-rec}\,(\mathit{flag},\ \mathit{child},\ \mathit{tree})$
$\quad = \quad \mathbf{if}\ \mathit{child} \in \text{nodes-rec}\,(\mathit{flag},\ \mathit{tree})$
$\quad\quad \mathbf{then\ if}\ ((\mathit{flag} = \texttt{'tree}) \land (\text{car}\,(\mathit{tree}) = \mathit{child}))$
$\quad\quad\quad \lor \quad ((\mathit{flag} \neq \texttt{'tree})$
$\quad\quad\quad\quad \land \quad (\mathit{child} \in \text{roots}\,(\mathit{tree})))\ \ \mathbf{then\ nil}$
$\quad\quad\quad \mathbf{else}\ \text{list}\,(\text{car}\,(\text{parent-rec}\,(\mathit{flag},\ \mathit{child},\ \mathit{tree})))\ \mathbf{endif}$
$\quad\quad \mathbf{else\ nil\ endif})$

EVENT: Disable setp-tree-unique-parent.

THEOREM: member-parent-parent
$(\text{proper-tree}\,(\mathit{flag},\ \mathit{tree})$
$\land\quad \text{setp}\,(\text{nodes-rec}\,(\mathit{flag},\ \mathit{tree}))$
$\land\quad (\mathit{parent} \in \text{parent-rec}\,(\mathit{flag},\ \mathit{child},\ \mathit{tree})))$
$\rightarrow\quad (\text{parent-rec}\,(\mathit{flag},\ \mathit{child},\ \mathit{tree}) = \text{list}\,(\mathit{parent}))$

THEOREM: parent-of-child
$(\text{proper-tree}\,(\mathit{flag},\ \mathit{tree})$
$\land\quad \text{setp}\,(\text{nodes-rec}\,(\mathit{flag},\ \mathit{tree}))$
$\land\quad (\mathit{child} \in \text{children-rec}\,(\mathit{flag},\ \mathit{parent},\ \mathit{tree})))$
$\rightarrow\quad (\text{parent-rec}\,(\mathit{flag},\ \mathit{child},\ \mathit{tree}) = \text{list}\,(\mathit{parent}))$

THEOREM: member-parent-member-tree
$(\mathit{parent} \in \text{parent-rec}\,(\mathit{flag},\ \mathit{child},\ \mathit{tree}))$
$\rightarrow\quad (\mathit{parent} \in \text{nodes-rec}\,(\mathit{flag},\ \mathit{tree}))$

THEOREM: node-that-has-child-is-in-tree
$\text{listp}\,(\text{children-rec}\,(\mathit{flag},\ \mathit{parent},\ \mathit{tree})) \rightarrow (\mathit{parent} \in \text{nodes-rec}\,(\mathit{flag},\ \mathit{tree}))$

THEOREM: node-that-has-parent-is-in-tree
$(\text{proper-tree}\,(\mathit{flag},\ \mathit{tree}) \land \text{listp}\,(\text{parent-rec}\,(\mathit{flag},\ \mathit{child},\ \mathit{tree})))$
$\rightarrow\quad (\mathit{child} \in \text{nodes-rec}\,(\mathit{flag},\ \mathit{tree}))$

THEOREM: sublistp-children-generalized
$(\text{proper-tree}\,(\mathit{flag},\ \mathit{tree})$
$\land\quad \text{sublistp}\,(\mathit{children},\ \text{children-rec}\,(\mathit{flag},\ \mathit{parent},\ \mathit{tree})))$
$\rightarrow\quad \text{sublistp}\,(\mathit{children},\ \text{nodes-rec}\,(\mathit{flag},\ \mathit{tree}))$

EVENT: Disable sublistp-children-generalized.

THEOREM: sublistp-children
$\text{proper-tree}\,(\mathit{flag},\ \mathit{tree})$
$\rightarrow\quad \text{sublistp}\,(\text{children-rec}\,(\mathit{flag},\ \mathit{parent},\ \mathit{tree}),\ \text{nodes-rec}\,(\mathit{flag},\ \mathit{tree}))$

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$(\mathit{flag}, \mathit{tree})$
$=$ **if** listp(tree)
then if $\mathit{flag} = $ `'tree`
then cons$(\mathit{tree}$, subtrees(`'forest`, cdr$(\mathit{tree})))$
else append(subtrees(`'tree`, car(tree)),
subtrees(`'forest`, cdr(tree))) **endif**
else nil endif

THEOREM: subtreep-subtrees
$(\mathit{subtree} \in \text{subtrees}(\mathit{flag}, \mathit{tree})) \rightarrow \text{subtreep}(\mathit{flag}, \mathit{subtree}, \mathit{tree})$

DEFINITION:
next-level$(\mathit{subtrees})$
$=$ **if** listp$(\mathit{subtrees})$
then append(cdar$(\mathit{subtrees})$, next-level(cdr$(\mathit{subtrees})$))
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
$\text{listp}(\mathit{subtrees}) \rightarrow (\text{count}(\text{next-level}(\mathit{subtrees})) < \text{count}(\mathit{subtrees}))$

THEOREM: next-level-of-tree-in-subtrees
proper-tree(`'forest`, *forest*)
$\rightarrow$ sublistp(*forest*, subtrees(`'forest`, *forest*))

THEOREM: subtrees-of-subtree-in-complete-subtrees
(proper-tree(`'tree`, *subtree*) $\land$ $(\mathit{subtree} \in \text{subtrees}(\mathit{flag}, \mathit{tree}))$)
$\rightarrow$ sublistp(subtrees(`'tree`, *subtree*), subtrees$(\mathit{flag}, \mathit{tree})$)

THEOREM: subtrees-of-subtrees-in-complete-subtrees
(proper-tree(`'forest`, *subtrees*) $\land$ sublistp(*subtrees*, subtrees$(\mathit{flag}, \mathit{tree})$))
$\rightarrow$ sublistp(subtrees(`'forest`, *subtrees*), subtrees$(\mathit{flag}, \mathit{tree})$)

THEOREM: next-level-in-subtrees-forest
proper-tree(`'forest`, *subtrees*)
$\rightarrow$ sublistp(next-level(*subtrees*), subtrees(`'forest`, *subtrees*))

THEOREM: next-level-of-subtrees-in-complete-subtrees
(proper-tree(`'forest`, *subtrees*) $\land$ sublistp(*subtrees*, subtrees$(\mathit{flag}, \mathit{tree})$))
$\rightarrow$ sublistp(next-level(*subtrees*), subtrees$(\mathit{flag}, \mathit{tree})$)

THEOREM: proper-tree-of-append
(proper-tree ('forest, a) $\land$ proper-tree ('forest, b))
$\rightarrow$ proper-tree ('forest, append (a, b))

THEOREM: proper-tree-next-level-of-proper-tree
proper-tree ('forest, *subtrees*)
$\rightarrow$ proper-tree ('forest, next-level (*subtrees*))

THEOREM: not-member-subtrees
(*root* $\notin$ nodes-rec (*flag*, *tree*))
$\rightarrow$ (cons (*root*, *forest*) $\notin$ subtrees (*flag*, *tree*))

THEOREM: not-member-no-children
(*parent* $\notin$ nodes-rec (*flag*, *tree*))
$\rightarrow$ (children-rec (*flag*, *parent*, *tree*) = **nil**)

THEOREM: no-children-in-rest-of-forest
(setp (append (nodes-rec ('tree, *tree*), nodes-rec ('forest, *forest*)))
$\land$ (*parent* $\in$ nodes-rec ('tree, *tree*)))
$\rightarrow$ (children-rec ('forest, *parent*, *forest*) = **nil**)

THEOREM: no-children-in-rest-of-tree
(setp (append (nodes-rec ('tree, *tree*), nodes-rec ('forest, *forest*)))
$\land$ (*parent* $\in$ nodes-rec ('forest, *forest*)))
$\rightarrow$ (children-rec ('tree, *parent*, *tree*) = **nil**)

THEOREM: member-subtree-member-tree
(cons (*root*, *forest*) $\in$ subtrees (*flag*, *tree*))
$\rightarrow$ (*root* $\in$ nodes-rec (*flag*, *tree*))

THEOREM: children-of-setp-tree
(setp (nodes-rec (*flag*, *tree*))
$\land$ proper-tree (*flag*, *tree*)
$\land$ (cons (*root*, *forest*) $\in$ subtrees (*flag*, *tree*)))
$\rightarrow$ (children-rec (*flag*, *root*, *tree*) = roots (*forest*))

THEOREM: node-has-parent
((*node* $\in$ nodes-rec (*flag*, *tree*))
$\land$ proper-tree (*flag*, *tree*)
$\land$ **if** *flag* = 'tree **then** *node* $\neq$ car (*tree*)
else *node* $\notin$ roots (*tree*) **endif**)
$\rightarrow$ (car (parent-rec (*flag*, *node*, *tree*)) $\in$ nodes-rec (*flag*, *tree*))

THEOREM: parent-is-not-itself-generalized
(setp (nodes-rec (*flag*, *tree*))
$\land$ proper-tree (*flag*, *tree*)
$\land$ listp (parent-rec (*flag*, *child*, *tree*)))
$\rightarrow$ (*child* $\neq$ car (parent-rec (*flag*, *child*, *tree*)))

THEOREM: parent-is-not-itself
$(\text{setp}\,(\text{nodes-rec}\,(\texttt{'tree},\ \textit{tree}))$
$\land\quad \text{proper-tree}\,(\texttt{'tree},\ \textit{tree})$
$\land\quad (\textit{child} \in \text{cdr}\,(\text{nodes-rec}\,(\texttt{'tree},\ \textit{tree}))))$
$\rightarrow\quad (\textit{child} \neq \text{car}\,(\text{parent-rec}\,(\texttt{'tree},\ \textit{child},\ \textit{tree})))$

THEOREM: listp-parent-rec-equals
$(\text{setp}\,(\text{nodes-rec}\,(\textit{flag},\ \textit{tree})) \land \text{proper-tree}\,(\textit{flag},\ \textit{tree}))$
$\rightarrow\quad (\text{listp}\,(\text{parent-rec}\,(\textit{flag},\ \textit{child},\ \textit{tree}))$
$\quad = \quad ((\textit{child} \in \text{nodes-rec}\,(\textit{flag},\ \textit{tree}))$
$\quad\quad \land \quad \textbf{if } \textit{flag} = \texttt{'tree} \textbf{ then } \textit{child} \neq \text{car}\,(\textit{tree})$
$\quad\quad\quad \textbf{else } \textit{child} \notin \text{roots}\,(\textit{tree}) \textbf{ endif}))$

THEOREM: parent-is-not-child
$(\text{setp}\,(\text{nodes-rec}\,(\textit{flag},\ \textit{tree}))$
$\land\quad \text{proper-tree}\,(\textit{flag},\ \textit{tree})$
$\land\quad \text{listp}\,(\text{parent-rec}\,(\textit{flag},\ \textit{child},\ \textit{tree})))$
$\rightarrow\quad (\text{car}\,(\text{parent-rec}\,(\textit{flag},\ \textit{child},\ \textit{tree})) \notin \text{children-rec}\,(\textit{flag},\ \textit{child},\ \textit{tree}))$

THEOREM: parent-not-in-children
$(\text{setp}\,(\text{nodes-rec}\,(\texttt{'tree},\ \textit{tree}))$
$\land\quad \text{proper-tree}\,(\texttt{'tree},\ \textit{tree})$
$\land\quad (\textit{parent} \in \text{cdr}\,(\text{nodes-rec}\,(\texttt{'tree},\ \textit{tree}))))$
$\rightarrow\quad (\textit{parent} \notin \text{children-rec}\,(\texttt{'tree},\ \textit{parent},\ \textit{tree}))$

```
;;; Variables and channel operations
```

DEFINITION: $\text{value}\,(\textit{key},\ \textit{state}) = \text{cdr}\,(\text{assoc}\,(\textit{key},\ \textit{state}))$

DEFINITION: $\text{channel}\,(\textit{name},\ \textit{state}) = \text{value}\,(\textit{name},\ \textit{state})$

DEFINITION:
$\text{empty}\,(\textit{name},\ \textit{state}) = (\neg\ \text{listp}\,(\text{channel}\,(\textit{name},\ \textit{state})))$

DEFINITION: $\text{head}\,(\textit{name},\ \textit{state}) = \text{car}\,(\text{channel}\,(\textit{name},\ \textit{state}))$

DEFINITION:
$\text{send}\,(\textit{channel},\ \textit{message},\ \textit{state})$
$=\quad \text{append}\,(\text{channel}\,(\textit{channel},\ \textit{state}),\ \text{list}\,(\textit{message}))$

DEFINITION:
$\text{receive}\,(\textit{channel},\ \textit{state}) = \text{cdr}\,(\text{channel}\,(\textit{channel},\ \textit{state}))$

```
;;; Program Specific
```

DEFINITION:
status (*node*, *state*) = value (cons (`'status`, *node*), *state*)

DEFINITION:
found-value (*node*, *state*) = value (cons (`'found-value`, *node*), *state*)

DEFINITION:
outstanding (*node*, *state*) = value (cons (`'outstanding`, *node*), *state*)

DEFINITION:
node-value (*node*, *state*) = value (cons (`'node-value`, *node*), *state*)

DEFINITION:
send-find (*to-children*, *old*, *new*)
= **if** listp (*to-children*)
  then (channel (car (*to-children*), *new*)
      = send (car (*to-children*), `'find`, *old*))
    ∧ send-find (cdr (*to-children*), *old*, *new*)
  else t endif

```
;;; The four program statements
```

DEFINITION:
receive-find (*old*, *new*, *node*, *from-parent*, *to-parent*, *to-children*)
= **if** head (*from-parent*, *old*) = `'find`
  then (channel (*from-parent*, *new*) = receive (*from-parent*, *old*))
    ∧ (status (*node*, *new*) = `'started`)
    ∧ (found-value (*node*, *new*) = node-value (*node*, *old*))
    ∧ (outstanding (*node*, *new*) = length (*to-children*))
    ∧ send-find (*to-children*, *old*, *new*)
    ∧ (channel (*to-parent*, *new*)
      = **if** length (*to-children*) $\simeq$ `0`
        then send (*to-parent*, node-value (*node*, *old*), *old*)
        else channel (*to-parent*, *old*) **endif**)
    ∧ changed (*old*,
        new,
        append (list (*from-parent*,
            to-parent,
            cons (`'status`, *node*),
            cons (`'found-value`, *node*),
            cons (`'outstanding`, *node*)),
          to-children))
  else changed (*old*, *new*, **nil**) **endif**

DEFINITION:

```
min (x, y)
=   if x < y  then fix (x)
    else fix (y) endif
```

DEFINITION:

```
receive-report (old, new, node, from-child, to-parent)
=   if empty (from-child, old)  then changed (old, new, nil)
    else (channel (from-child, new) = receive (from-child, old))
           ∧   (found-value (node, new)
                 =   min (found-value (node, old),
                          head (from-child, old)))
           ∧   (outstanding (node, new)
                 =   (outstanding (node, old) − 1))
           ∧   (channel (to-parent, new)
                 =   if outstanding (node, new) ≃ 0
                     then send (to-parent,
                                found-value (node, new),
                                old)
                     else channel (to-parent, old) endif)
           ∧   changed (old,
                        new,
                        list (from-child,
                              to-parent,
                              cons ('outstanding, node),
                              cons ('found-value, node))) endif
```

DEFINITION:

```
start (old, new, root, to-children)
=   if status (root, old) = 'not-started
    then (status (root, new) = 'started)
           ∧   (found-value (root, new) = node-value (root, old))
           ∧   (outstanding (root, new) = length (to-children))
           ∧   send-find (to-children, old, new)
           ∧   changed (old,
                        new,
                        append (list (cons ('status, root),
                                      cons ('found-value, root),
                                      cons ('outstanding, root)),
                                to-children))
    else changed (old, new, nil) endif
```

DEFINITION:

```
root-receive-report (old, new, root, from-child)
=   if empty (from-child, old)  then changed (old, new, nil)
```

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*))
= ((car (*statement*) = `'receive-find`)
  ∧ (cadr (*statement*) ∈ *nodes*)
  ∧ 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**))

DEFINITION:
rrp (*node*, *children*, *parent*)
= **if** listp (*children*)
then cons (list (`'receive-report`,
node,
cons (car (*children*), *node*),
cons (*node*, *parent*)),
rrp (*node*, cdr (*children*), *parent*))
else nil endif

THEOREM: member-rrp
(*statement* ∈ rrp (*node*, *children*, *parent*))
= ((car (*statement*) = `'receive-report`)
∧ (cadr (*statement*) = *node*)
∧ listp (caddr (*statement*))
∧ (caaddr (*statement*) ∈ *children*)
∧ (cdaddr (*statement*) = *node*)
∧ listp (cadddr (*statement*))
∧ (caadddr (*statement*) = *node*)
∧ (cdadddr (*statement*) = *parent*)
∧ (cddddr (*statement*) = **nil**))

DEFINITION:
receive-report-prg (*nodes*, *tree*)
= **if** listp (*nodes*)
then append (rrp (car (*nodes*),
children (car (*nodes*), *tree*),
parent (car (*nodes*), *tree*)),
receive-report-prg (cdr (*nodes*), *tree*))
else nil endif

THEOREM: member-receive-report-prg
(*statement* ∈ receive-report-prg (*nodes*, *tree*))
= ((car (*statement*) = `'receive-report`)
∧ (cadr (*statement*) ∈ *nodes*)
∧ 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*))
∧ (cddddr (*statement*) = **nil**))

DEFINITION:
start-prg (*root*, *tree*)
= list (list (`'start`, *root*, rfp (*root*, children (*root*, *tree*))))

Theorem: member-start-prg
$(\textit{statement} \in \text{start-prg}\,(\textit{root},\ \textit{tree}))$
$= \quad ((\text{car}\,(\textit{statement}) = \texttt{'start})$
$\quad \land \quad (\text{cadr}\,(\textit{statement}) = \textit{root})$
$\quad \land \quad (\text{caddr}\,(\textit{statement}) = \text{rfp}\,(\textit{root}, \text{children}\,(\textit{root},\ \textit{tree})))$
$\quad \land \quad (\text{cdddr}\,(\textit{statement}) = \textbf{nil}))$

Definition:
rrrp (*root*, *children*)
= **if** listp (*children*)
then cons (list (`'root-receive-report`,
root,
cons (car (*children*), *root*)),
rrrp (*root*, cdr (*children*)))
else nil endif

Theorem: member-rrrp
$(\textit{statement} \in \text{rrrp}\,(\textit{root},\ \textit{children}))$
$= \quad ((\text{car}\,(\textit{statement}) = \texttt{'root-receive-report})$
$\quad \land \quad (\text{cadr}\,(\textit{statement}) = \textit{root})$
$\quad \land \quad \text{listp}\,(\text{caddr}\,(\textit{statement}))$
$\quad \land \quad (\text{caaddr}\,(\textit{statement}) \in \textit{children})$
$\quad \land \quad (\text{cdaddr}\,(\textit{statement}) = \textit{root})$
$\quad \land \quad (\text{cdddr}\,(\textit{statement}) = \textbf{nil}))$

Definition:
root-receive-report-prg (*root*, *tree*) = rrrp (*root*, children (*root*, *tree*))

Theorem: member-root-receive-report-prg
$(\textit{statement} \in \text{root-receive-report-prg}\,(\textit{root},\ \textit{tree}))$
$= \quad ((\text{car}\,(\textit{statement}) = \texttt{'root-receive-report})$
$\quad \land \quad (\text{cadr}\,(\textit{statement}) = \textit{root})$
$\quad \land \quad \text{listp}\,(\text{caddr}\,(\textit{statement}))$
$\quad \land \quad (\text{caaddr}\,(\textit{statement}) \in \text{children}\,(\textit{root},\ \textit{tree}))$
$\quad \land \quad (\text{cdaddr}\,(\textit{statement}) = \textit{root})$
$\quad \land \quad (\text{cdddr}\,(\textit{statement}) = \textbf{nil}))$

Definition:
tree-prg (*tree*)
= append (start-prg (car (*tree*), *tree*),
append (root-receive-report-prg (car (*tree*), *tree*),
append (receive-find-prg (cdr (nodes (*tree*)), *tree*),
receive-report-prg (cdr (nodes (*tree*)), *tree*))))

Theorem: equal-if

$$
\begin{array}{l}
(\textbf{if } \mathit{test} \textbf{ then } \mathit{p1} \\
\;\textbf{else } \mathit{p2} \textbf{ endif} \\
\;= \quad \textbf{if } \mathit{test} \textbf{ then } \mathit{r1} \\
\qquad\;\; \textbf{else } \mathit{r2} \textbf{ endif}) \\
= \quad \textbf{if } \mathit{test} \textbf{ then } \mathit{p1} = \mathit{r1} \\
\qquad \textbf{else } \mathit{p2} = \mathit{r2} \textbf{ endif}
\end{array}
$$

THEOREM: member-tree-prg

$$
\begin{array}{l}
(\mathit{statement} \in \text{tree-prg}(\mathit{tree})) \\
= \quad (((\text{car}(\mathit{statement}) = \texttt{'start}) \\
\qquad \land \quad (\text{cadr}(\mathit{statement}) = \text{car}(\mathit{tree})) \\
\qquad \land \quad (\text{caddr}(\mathit{statement}) \\
\qquad\qquad = \quad \text{rfp}(\text{car}(\mathit{tree}),\ \text{children}(\text{car}(\mathit{tree}),\ \mathit{tree}))) \\
\qquad \land \quad (\text{cdddr}(\mathit{statement}) = \textbf{nil})) \\
\quad \lor \quad ((\text{car}(\mathit{statement}) = \texttt{'root-receive-report}) \\
\qquad\quad \land \quad (\text{cadr}(\mathit{statement}) = \text{car}(\mathit{tree})) \\
\qquad\quad \land \quad \text{listp}(\text{caddr}(\mathit{statement})) \\
\qquad\quad \land \quad (\text{caaddr}(\mathit{statement}) \in \text{children}(\text{car}(\mathit{tree}),\ \mathit{tree})) \\
\qquad\quad \land \quad (\text{cdaddr}(\mathit{statement}) = \text{car}(\mathit{tree})) \\
\qquad\quad \land \quad (\text{cdddr}(\mathit{statement}) = \textbf{nil})) \\
\quad \lor \quad ((\text{car}(\mathit{statement}) = \texttt{'receive-find}) \\
\qquad\quad \land \quad (\text{cadr}(\mathit{statement}) \in \text{cdr}(\text{nodes}(\mathit{tree}))) \\
\qquad\quad \land \quad \text{listp}(\text{caddr}(\mathit{statement})) \\
\qquad\quad \land \quad (\text{caaddr}(\mathit{statement}) = \text{parent}(\text{cadr}(\mathit{statement}),\ \mathit{tree})) \\
\qquad\quad \land \quad (\text{cdaddr}(\mathit{statement}) = \text{cadr}(\mathit{statement})) \\
\qquad\quad \land \quad \text{listp}(\text{cadddr}(\mathit{statement})) \\
\qquad\quad \land \quad (\text{caadddr}(\mathit{statement}) = \text{cadr}(\mathit{statement})) \\
\qquad\quad \land \quad (\text{cdadddr}(\mathit{statement}) = \text{parent}(\text{cadr}(\mathit{statement}),\ \mathit{tree})) \\
\qquad\quad \land \quad (\text{caddddr}(\mathit{statement}) \\
\qquad\qquad\quad = \quad \text{rfp}(\text{cadr}(\mathit{statement}), \\
\qquad\qquad\qquad\quad \text{children}(\text{cadr}(\mathit{statement}),\ \mathit{tree}))) \\
\qquad\quad \land \quad (\text{cdddddr}(\mathit{statement}) = \textbf{nil})) \\
\quad \lor \quad ((\text{car}(\mathit{statement}) = \texttt{'receive-report}) \\
\qquad\quad \land \quad (\text{cadr}(\mathit{statement}) \in \text{cdr}(\text{nodes}(\mathit{tree}))) \\
\qquad\quad \land \quad \text{listp}(\text{caddr}(\mathit{statement})) \\
\qquad\quad \land \quad (\text{caaddr}(\mathit{statement}) \in \text{children}(\text{cadr}(\mathit{statement}),\ \mathit{tree})) \\
\qquad\quad \land \quad (\text{cdaddr}(\mathit{statement}) = \text{cadr}(\mathit{statement})) \\
\qquad\quad \land \quad \text{listp}(\text{cadddr}(\mathit{statement})) \\
\qquad\quad \land \quad (\text{caadddr}(\mathit{statement}) = \text{cadr}(\mathit{statement})) \\
\qquad\quad \land \quad (\text{cdadddr}(\mathit{statement}) = \text{parent}(\text{cadr}(\mathit{statement}),\ \mathit{tree})) \\
\qquad\quad \land \quad (\text{cddddr}(\mathit{statement}) = \textbf{nil})))
\end{array}
$$

```
;;; Correctness
```

DEFINITION:
treep (*tree*)
= (setp (nodes (*tree*))
 ∧ all-numberps (nodes (*tree*))
 ∧ proper-tree (`'tree`, *tree*))

DEFINITION:
total-outstanding (*nodes*, *tree*, *state*)
= **if** listp (*nodes*)
 then total-outstanding (cdr (*nodes*), *tree*, *state*)
 + **if** status (car (*nodes*), *state*) = `'started`
 then outstanding (car (*nodes*), *state*)
 else 1 + length (children (car (*nodes*), *tree*)) **endif**
 else `0` **endif**

DEFINITION:
dl (*down-links*, *state*)
= **if** listp (*down-links*)
 then ((empty (car (*down-links*), *state*)
 ∧ (status (caar (*down-links*), *state*)
 = status (cdar (*down-links*), *state*)))
 ∨ ((channel (car (*down-links*), *state*) = list (`'find`))
 ∧ (status (caar (*down-links*), *state*) = `'started`)
 ∧ (status (cdar (*down-links*), *state*)
 = `'not-started`)))
 ∧ dl (cdr (*down-links*), *state*)
 else t endif

DEFINITION:
done (*node*, *state*)
= ((status (*node*, *state*) = `'started`)
 ∧ (outstanding (*node*, *state*) $\simeq$ `0`))

DEFINITION:
ul (*up-links*, *state*)
= **if** listp (*up-links*)
 then (empty (car (*up-links*), *state*)
 ∨ ((channel (car (*up-links*), *state*)
 = list (found-value (caar (*up-links*), *state*)))
 ∧ done (caar (*up-links*), *state*)))
 ∧ ul (cdr (*up-links*), *state*)
 else t endif

DEFINITION:
reported (*node*, *parent*, *state*)
= (done (*node*, *state*) ∧ empty (cons (*node*, *parent*), *state*))

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 (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
(all-empty ($channels$, $state$) $\wedge$ ($channel \in channels$))
$\rightarrow$ ($\neg$ listp (channel ($channel$, $state$)))

THEOREM: not-started-implies-not-started
(not-started ($nodes$, $state$) $\wedge$ ($node \in nodes$))
$\rightarrow$ (cdr (assoc (cons (`'status`, $node$), $state$)) = `'not-started`)

THEOREM: all-empty-append
all-empty (append (a, b), $state$)
= (all-empty (a, $state$) $\wedge$ all-empty (b, $state$))

THEOREM: all-empty-implies-ul
all-empty ($up\text{-}links$, $state$) $\rightarrow$ ul ($up\text{-}links$, $state$)

DEFINITION:
nodes-in-channels ($channels$)
= **if** listp ($channels$)
then cons (caar ($channels$),
cons (cdar ($channels$), nodes-in-channels (cdr ($channels$))))
else nil endif

THEOREM: all-empty-not-started-implies-dl
(all-empty ($down\text{-}links$, $state$)
$\wedge$ not-started (nodes-in-channels ($down\text{-}links$), $state$))
$\rightarrow$ dl ($down\text{-}links$, $state$)

THEOREM: not-started-implies-no
not-started ($nodes$, $state$) $\rightarrow$ no ($nodes$, $tree$, $state$)

THEOREM: nodes-in-down-links-1-in-nodes
($node \in$ nodes-in-channels (down-links-1 ($parent$, $children$)))
= **if** listp ($children$) **then** $node \in$ cons ($parent$, $children$)
else f endif

THEOREM: nodes-in-channels-append
nodes-in-channels (append (a, b))
= append (nodes-in-channels (a), nodes-in-channels (b))

THEOREM: nodes-in-down-links-in-nodes
(proper-tree (`'tree`, $tree$)
$\wedge$ ($node \in$ nodes-in-channels (down-links ($nodes$, $tree$))))
$\rightarrow$ ($node \in$ nodes ($tree$))

THEOREM: sublistp-not-started
(sublistp (*sub*, *list*) $\wedge$ not-started (*list*, *state*)) $\rightarrow$ not-started (*sub*, *state*)

THEOREM: sublistp-down-links-1
(sublistp (*children*, *nodes*) $\wedge$ (*parent* $\in$ *nodes*))
$\rightarrow$ sublistp (nodes-in-channels (down-links-1 (*parent*, *children*)), *nodes*)

THEOREM: children-of-non-node
(*parent* $\notin$ nodes-rec (*flag*, *tree*))
$\rightarrow$ (children-rec (*flag*, *parent*, *tree*) = **nil**)

THEOREM: down-links-is-sublistp
proper-tree (`'tree`, *tree*)
$\rightarrow$ sublistp (nodes-in-channels (down-links (*nodes*, *tree*)),
nodes-rec (`'tree`, *tree*))

THEOREM: initial-conditions-imply-invariant
(proper-tree (`'tree`, *tree*)
$\wedge$ all-empty (all-channels (*tree*), *state*)
$\wedge$ not-started (nodes (*tree*), *state*))
$\rightarrow$ 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*))),
state,
node-value (caar (*subtrees*), *state*)))
$\wedge$ 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*) $\wedge$ 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

$$
\begin{array}{l}
(\text{no}\,(\textit{nodes},\ \textit{tree},\ \textit{state}) \\
\ \ \wedge\quad (\textit{node} \in \textit{nodes}) \\
\ \ \wedge\quad (\text{status}\,(\textit{node},\ \textit{state}) = \texttt{'started})) \\
\to\quad ((\text{number-not-reported}\,(\text{children}\,(\textit{node},\ \textit{tree}),\ \textit{node},\ \textit{state}) \\
\qquad\quad = \quad \text{cdr}\,(\text{assoc}\,(\text{cons}\,(\texttt{'outstanding},\ \textit{node}),\ \textit{state}))) \\
\qquad\ \ \wedge\quad (\text{cdr}\,(\text{assoc}\,(\text{cons}\,(\texttt{'outstanding},\ \textit{node}),\ \textit{state})) \in \mathbf{N}) \\
\qquad\ \ \wedge\quad (\text{cdr}\,(\text{assoc}\,(\text{cons}\,(\texttt{'found-value},\ \textit{node}),\ \textit{state})) \\
\qquad\qquad = \quad \text{min-of-reported}\,(\text{children}\,(\textit{node},\ \textit{tree}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \textit{node}, \\
\qquad\qquad\qquad\qquad\qquad\qquad \textit{state}, \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{node-value}\,(\textit{node},\ \textit{state}))))
\end{array}
$$

Theorem: total-outstanding-0-implies

$$
\begin{array}{l}
(((\text{total-outstanding}\,(\textit{nodes},\ \textit{tree},\ \textit{state}) = \texttt{0}) \\
\quad \wedge\quad (\textit{node} \in \textit{nodes}) \\
\quad \wedge\quad (\text{cdr}\,(\text{assoc}\,(\text{cons}\,(\texttt{'outstanding},\ \textit{node}),\ \textit{state})) \in \mathbf{N})) \\
\quad \to\quad (\text{cdr}\,(\text{assoc}\,(\text{cons}\,(\texttt{'outstanding},\ \textit{node}),\ \textit{state})) = \texttt{0})) \\
\wedge\quad (((\text{total-outstanding}\,(\textit{nodes},\ \textit{tree},\ \textit{state}) = \texttt{0}) \wedge (\textit{node} \in \textit{nodes})) \\
\qquad \to\quad (\text{cdr}\,(\text{assoc}\,(\text{cons}\,(\texttt{'status},\ \textit{node}),\ \textit{state})) = \texttt{'started}))
\end{array}
$$

Theorem: number-not-reported-0-implies

$$
\begin{array}{l}
((\text{number-not-reported}\,(\textit{children},\ \textit{parent},\ \textit{state}) = \texttt{0}) \wedge (\textit{node} \in \textit{children})) \\
\to\quad \text{reported}\,(\textit{node},\ \textit{parent},\ \textit{state})
\end{array}
$$

Theorem: proper-tree-tree-implies-nodes-exists

$$
\text{proper-tree}\,(\texttt{'tree},\ \textit{tree}) \to \text{listp}\,(\text{nodes-rec}\,(\texttt{'tree},\ \textit{tree}))
$$

Theorem: min-of-two-nodes-values

$$
\begin{array}{l}
\text{min}\,(\text{min-node-value}\,(\textit{forest-1}, \\
\qquad\qquad\qquad\qquad \textit{state}, \\
\qquad\qquad\qquad\qquad \text{cdr}\,(\text{assoc}\,(\text{cons}\,(\texttt{'node-value},\ \textit{root}),\ \textit{state}))), \\
\qquad \text{min-node-value}\,(\textit{rest-of-forest},\ \textit{state},\ \textit{min})) \\
= \quad \text{min-node-value}\,(\text{cons}\,(\textit{root},\ \text{append}\,(\textit{forest-1},\ \textit{rest-of-forest})),\ \textit{state},\ \textit{min})
\end{array}
$$

Theorem: found-value-min-value-generalized

$$
\begin{array}{l}
(\text{found-value-node-value}\,(\textit{forest},\ \textit{state}) \\
\ \ \wedge\quad (\text{number-not-reported}\,(\text{roots}\,(\textit{forest}),\ \textit{root},\ \textit{state}) = \texttt{0}) \\
\ \ \wedge\quad \text{proper-tree}\,(\texttt{'forest},\ \textit{forest})) \\
\to\quad (\text{min-of-reported}\,(\text{roots}\,(\textit{forest}),\ \textit{root},\ \textit{state},\ \textit{min}) \\
\qquad\quad = \quad \text{min-node-value}\,(\text{nodes-rec}\,(\texttt{'forest},\ \textit{forest}),\ \textit{state},\ \textit{min}))
\end{array}
$$

Theorem: no-at-termination

$$
\begin{array}{l}
(\text{proper-tree}\,(\texttt{'tree},\ \textit{tree}) \\
\ \ \wedge\quad \text{proper-tree}\,(\texttt{'forest},\ \textit{subtrees})
\end{array}
$$

```
 ∧   setp (nodes-rec ('tree, tree))
 ∧   no (nodes-rec ('tree, tree), tree, state)
 ∧   (total-outstanding (nodes-rec ('tree, tree), tree, state) = 0)
 ∧   sublistp (subtrees, subtrees ('tree, tree)))
→    found-value-node-value (subtrees, state)
```

THEOREM: inv-implies-augmented-correctness-condition

```
(proper-tree ('tree, tree)
 ∧   setp (nodes-rec ('tree, tree))
 ∧   inv (tree, state)
 ∧   (total-outstanding (nodes (tree), tree, state) = 0))
→    correct (tree, state)
```

DEFINITION:

```
send-find-func (to-children, old)
=   if listp (to-children)
    then update-assoc (car (to-children),
                       send (car (to-children), 'find, old),
                       send-find-func (cdr (to-children), old))
    else old endif
```

DEFINITION:

```
receive-find-func (old, node, from-parent, to-parent, to-children)
=   if head (from-parent, old) = 'find
    then update-assoc (from-parent,
                       receive (from-parent, old),
                       update-assoc (cons ('status, node),
                                     'started,
                                     update-assoc (cons ('found-value,
                                                         node),
                                                   node-value (node, old),
                                                   update-assoc (cons ('outstanding,
                                                                       node),
                                                                 length (to-children),
                                                                 if length (to-children) ≃ 0
                                                                 then update-assoc (to-parent,
                                                                                    send (to-parent,
                                                                                          node-value (node
                                                                                                      old),
                                                                                          old),
                                                                                    send-find-func (to-chil
                                                                                                    old))
                                                                 else send-find-func (to-children,
                                                                                      old) endif))))
    else 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
$(\text{all-numberps}\,(\text{nodes-rec}\,(\mathit{flag},\ \mathit{tree})) \land (\mathit{node} \in \text{nodes-rec}\,(\mathit{flag},\ \mathit{tree})))$
$\rightarrow \quad ((\text{pack}\,(x) = \mathit{node}) = \mathbf{f})$

THEOREM: parent-is-not-a-litatom
$(\text{all-numberps}\,(\text{nodes-rec}\,(\texttt{'tree},\ \mathit{tree}))$
$\land \quad \text{setp}\,(\text{nodes-rec}\,(\texttt{'tree},\ \mathit{tree}))$
$\land \quad \text{proper-tree}\,(\texttt{'tree},\ \mathit{tree})$
$\land \quad (\mathit{child} \in \text{cdr}\,(\text{nodes-rec}\,(\texttt{'tree},\ \mathit{tree}))))$
$\rightarrow \quad ((\text{pack}\,(x) = \text{car}\,(\text{parent-rec}\,(\texttt{'tree},\ \mathit{child},\ \mathit{tree}))) = \mathbf{f})$

THEOREM: children-are-not-litatoms
$(\text{all-numberps}\,(\text{nodes-rec}\,(\mathit{flag},\ \mathit{tree}))$
$\land \quad \text{proper-tree}\,(\mathit{flag},\ \mathit{tree})$
$\land \quad (\mathit{child} \in \text{children-rec}\,(\mathit{flag},\ \mathit{parent},\ \mathit{tree})))$
$\rightarrow \quad ((\text{pack}\,(x) = \mathit{child}) = \mathbf{f})$

THEOREM: children-are-not-litatoms-member
$(\text{all-numberps}\,(\text{nodes-rec}\,(\mathit{flag},\ \mathit{tree})) \land \text{proper-tree}\,(\mathit{flag},\ \mathit{tree}))$
$\rightarrow \quad ((\text{pack}\,(x) \in \text{children-rec}\,(\mathit{flag},\ \mathit{parent},\ \mathit{tree})) = \mathbf{f})$

THEOREM: send-find-of-update-assoc
$(\mathit{key} \notin \mathit{to\text{-}children})$
$\rightarrow \quad (\text{send-find}\,(\mathit{to\text{-}children},\ \mathit{old},\ \text{update-assoc}\,(\mathit{key},\ \mathit{value},\ \mathit{state}))$
$\qquad = \quad \text{send-find}\,(\mathit{to\text{-}children},\ \mathit{old},\ \mathit{state}))$

THEOREM: assoc-of-send-find-func
$(\mathit{key} \notin \mathit{to\text{-}children})$
$\rightarrow \quad (\text{assoc}\,(\mathit{key},\ \text{send-find-func}\,(\mathit{to\text{-}children},\ \mathit{old})) = \text{assoc}\,(\mathit{key},\ \mathit{old}))$

THEOREM: about-rfp
$(p \notin c) \rightarrow (\text{cons}\,(v,\ p) \notin \text{rfp}\,(v,\ c))$

THEOREM: about-rfp-numberp
$(a \in \mathbf{N}) \rightarrow (\text{cons}\,(\text{pack}\,(x),\ y) \notin \text{rfp}\,(a,\ b))$

THEOREM: parent-not-in-rfp
$(\text{setp}\,(\text{nodes-rec}\,(\texttt{'tree},\ \mathit{tree}))$
$\land \quad \text{proper-tree}\,(\texttt{'tree},\ \mathit{tree})$
$\land \quad (v \in \text{cdr}\,(\text{nodes-rec}\,(\texttt{'tree},\ \mathit{tree}))))$
$\rightarrow \quad (\text{cons}\,(v,\ \text{car}\,(\text{parent-rec}\,(\texttt{'tree},\ v,\ \mathit{tree})))$
$\qquad \notin \quad \text{rfp}\,(v,\ \text{children-rec}\,(\texttt{'tree},\ v,\ \mathit{tree})))$

THEOREM: to-node-not-in-rfp
$(node \notin children) \rightarrow (\text{cons}\,(x,\ node) \notin \text{rfp}\,(node,\ children))$

THEOREM: uc-of-send-find-func
sublistp (*to-children*, *excpt*)
$\rightarrow$ (uc (*old*, send-find-func (*to-children*, *state*), *keys*, *excpt*)
$=$ uc (*old*, *state*, *keys*, *excpt*))

THEOREM: receive-find-func-implements-receive-find
(treep (*tree*) $\wedge$ (*statement* $\in$ receive-find-prg (cdr (nodes (*tree*)), *tree*)))
$\rightarrow$ n (*old*,
receive-find-func (*old*,
cadr (*statement*),
caddr (*statement*),
cadddr (*statement*),
caddddr (*statement*)),
statement)

DEFINITION:
receive-report-func (*old*, *node*, *from-child*, *to-parent*)
$=$ **if** empty (*from-child*, *old*) **then** *old*
else update-assoc (*from-child*,
receive (*from-child*, *old*),
update-assoc (cons (`'found-value`, *node*),
min (found-value (*node*, *old*),
head (*from-child*, *old*)),
update-assoc (cons (`'outstanding`,
node),
outstanding (*node*,
old) $-$ 1,
if (outstanding (*node*,
old) $-$ 1) $\simeq$ `0`
then update-assoc (*to-parent*,
send (*to-parent*,
min (found-value (*node*,
old),
head (*from-child*,
old)),
old),
old)
else *old* **endif**))) **endif**

THEOREM: receive-report-func-implements-receive-report
(treep (*tree*) $\wedge$ (*statement* $\in$ receive-report-prg (cdr (nodes (*tree*)), *tree*)))
$\rightarrow$ n (*old*,

$$
\begin{array}{l}
\quad\text{receive-report-func}\,(old, \\
\qquad\qquad\qquad\text{cadr}\,(statement), \\
\qquad\qquad\qquad\text{caddr}\,(statement), \\
\qquad\qquad\qquad\text{cadddr}\,(statement)), \\
\quad statement)
\end{array}
$$

Definition:
start-func $(old,\ root,\ to\text{-}children)$

$$
\begin{array}{ll}
= & \textbf{if}\ \text{status}\,(root,\ old) = \texttt{'not-started} \\
& \textbf{then}\ \text{update-assoc}\,(\text{cons}\,(\texttt{'status},\ root), \\
& \qquad \texttt{'started}, \\
& \qquad \text{update-assoc}\,(\text{cons}\,(\texttt{'found-value},\ root), \\
& \qquad\qquad \text{node-value}\,(root,\ old), \\
& \qquad\qquad \text{update-assoc}\,(\text{cons}\,(\texttt{'outstanding}, \\
& \qquad\qquad\qquad root), \\
& \qquad\qquad\qquad \text{length}\,(to\text{-}children), \\
& \qquad\qquad\qquad \text{send-find-func}\,(to\text{-}children, \\
& \qquad\qquad\qquad\qquad old)))) \\
& \textbf{else}\ old\ \textbf{endif}
\end{array}
$$

Theorem: start-func-implements-start

$$
\begin{array}{l}
(\text{treep}\,(tree) \land (statement \in \text{start-prg}\,(\text{car}\,(tree),\ tree))) \\
\rightarrow\quad \text{n}\,(old,\ \text{start-func}\,(old,\ \text{cadr}\,(statement),\ \text{caddr}\,(statement)),\ statement)
\end{array}
$$

Definition:
root-receive-report-func $(old,\ root,\ from\text{-}child)$

$$
\begin{array}{ll}
= & \textbf{if}\ \text{empty}\,(from\text{-}child,\ old)\ \ \textbf{then}\ old \\
& \textbf{else}\ \text{update-assoc}\,(from\text{-}child, \\
& \qquad \text{receive}\,(from\text{-}child,\ old), \\
& \qquad \text{update-assoc}\,(\text{cons}\,(\texttt{'found-value},\ root), \\
& \qquad\qquad \text{min}\,(\text{found-value}\,(root,\ old), \\
& \qquad\qquad\quad \text{head}\,(from\text{-}child,\ old)), \\
& \qquad\qquad \text{update-assoc}\,(\text{cons}\,(\texttt{'outstanding}, \\
& \qquad\qquad\qquad root), \\
& \qquad\qquad\qquad \text{outstanding}\,(root, \\
& \qquad\qquad\qquad\qquad old) - 1, \\
& \qquad\qquad\qquad old)))\ \textbf{endif}
\end{array}
$$

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

$$
\begin{array}{l}
(\text{treep}\,(tree) \land (statement \in \text{root-receive-report-prg}\,(\text{car}\,(tree),\ tree))) \\
\rightarrow\quad \text{n}\,(old, \\
\qquad\quad \text{root-receive-report-func}\,(old,\ \text{cadr}\,(statement),\ \text{caddr}\,(statement)), \\
\qquad\quad statement)
\end{array}
$$

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*))
$\land$ treep (*tree*)
$\land$ (*node* $\in$ nodes (*tree*)))
$\to$ invariant (`‘(equal (node-value ’,node state) ’,k)`,
tree-prg (*tree*))

THEOREM: dl-implies-instance-of-dl
(dl (*down-links*, *state*) $\land$ (*down-link* $\in$ *down-links*))
$\to$ ((empty (*down-link*, *state*)
$\land$ (status (car (*down-link*), *state*)
$=$ status (cdr (*down-link*), *state*)))
$\lor$ ((channel (*down-link*, *state*) $=$ list (`’find`))
$\land$ (status (car (*down-link*), *state*) $=$ `’started`)
$\land$ (status (cdr (*down-link*), *state*) $=$ `’not-started`)))

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

THEOREM: ul-implies-instance-of-ul
(ul (*uplinks*, *state*) $\land$ (*uplink* $\in$ *uplinks*))
$\to$ (empty (*uplink*, *state*)
$\lor$ ((channel (*uplink*, *state*)
$=$ list (found-value (car (*uplink*), *state*)))
$\land$ done (car (*uplink*), *state*)))

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

THEOREM: ul-implies-instance-of-ul-not-empty-uplink
(ul (*uplinks*, *state*) $\land$ (*uplink* $\in$ *uplinks*) $\land$ ($\neg$ empty (*uplink*, *state*)))
$\to$ ((cdr (assoc (*uplink*, *state*)) $=$ list (found-value (car (*uplink*), *state*)))
$\land$ (cdr (assoc (cons (`’status`, car (*uplink*)), *state*))
$=$ `’started`)
$\land$ (cdr (assoc (cons (`’outstanding`, car (*uplink*)), *state*)) $\simeq$ `0`))

THEOREM: no-implies-instance-of-no
(no (*nodes*, *tree*, *state*)
 ∧ (*node* ∈ *nodes*)
 ∧ (status (*node*, *state*) = `'started`))
→ ((cdr (assoc (cons (`'outstanding`, *node*), *state*))
  = number-not-reported (children-rec (`'tree`, *node*, *tree*),
                node,
                state))
 ∧ (cdr (assoc (cons (`'found-value`, *node*), *state*))
  = min-of-reported (children-rec (`'tree`, *node*, *tree*),
                node,
                state,
                node-value (*node*, *state*))))

THEOREM: member-down-links-1
(*down-link* ∈ down-links-1 (*parent*, *children*))
= ((car (*down-link*) = *parent*)
 ∧ (cdr (*down-link*) ∈ *children*)
 ∧ listp (*down-link*))

THEOREM: member-down-links
(*down-link* ∈ down-links (*nodes*, *tree*))
= ((car (*down-link*) ∈ *nodes*)
 ∧ (cdr (*down-link*) ∈ children (car (*down-link*), *tree*))
 ∧ listp (*down-link*))

THEOREM: parent-not-child
(proper-tree (*flag*, *tree*) ∧ setp (nodes-rec (*flag*, *tree*)))
→ (*parent* ∉ children-rec (*flag*, *parent*, *tree*))

THEOREM: parent-not-grandchild
(proper-tree (*flag*, *tree*)
 ∧ setp (nodes-rec (*flag*, *tree*))
 ∧ (*child* ∈ children-rec (*flag*, *parent*, *tree*)))
→ (*parent* ∉ children-rec (*flag*, *child*, *tree*))

THEOREM: parent-of-parent-not-node
(proper-tree (*flag*, *tree*)
 ∧ setp (nodes-rec (*flag*, *tree*))
 ∧ listp (parent-rec (*flag*, *node*, *tree*))
 ∧ listp (parent-rec (*flag*, car (parent-rec (*flag*, *node*, *tree*)), *tree*)))
→ (car (parent-rec (*flag*, car (parent-rec (*flag*, *node*, *tree*)), *tree*)) ≠ *node*)

THEOREM: member-rfp
(*channel* ∈ rfp (*parent*, *children*))

$=\quad ((\mathrm{car}\,(\mathit{channel}) = \mathit{parent})$
$\quad\quad \land \quad (\mathrm{cdr}\,(\mathit{channel}) \in \mathit{children})$
$\quad\quad \land \quad \mathrm{listp}\,(\mathit{channel}))$

THEOREM: send-find-implies
$(\text{send-find}\,(\mathit{channels},\ \mathit{old},\ \mathit{new}) \land (\mathit{key} \in \mathit{channels}))$
$\rightarrow \quad (\mathrm{cdr}\,(\mathrm{assoc}\,(\mathit{key},\ \mathit{new})) = \mathrm{send}\,(\mathit{key},\ \texttt{'find},\ \mathit{old}))$

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

THEOREM: assoc-equal-cons
$(\mathrm{assoc}\,(\mathit{key},\ \mathit{alist}) = \mathrm{cons}\,(\mathit{key},\ \mathit{value}))$
$= \quad (\mathrm{listp}\,(\mathrm{assoc}\,(\mathit{key},\ \mathit{alist})) \land (\mathrm{cdr}\,(\mathrm{assoc}\,(\mathit{key},\ \mathit{alist})) = \mathit{value}))$

THEOREM: send-find-general
$(\text{send-find}\,(\mathit{channels},\ \mathit{old},\ \mathit{new}) \land (\mathit{key} \in \mathit{channels}))$
$\rightarrow \quad (\mathrm{assoc}\,(\mathit{key},\ \mathit{new}) = \mathrm{cons}\,(\mathit{key},\ \mathrm{send}\,(\mathit{key},\ \texttt{'find},\ \mathit{old})))$

THEOREM: all-numberps-do-not-contain-litatom
$\text{all-numberps}\,(\mathit{list}) \rightarrow (\mathrm{pack}\,(x) \notin \mathit{list})$

THEOREM: all-numberps-append
$\text{all-numberps}\,(\mathrm{append}\,(x,\ y)) = (\text{all-numberps}\,(x) \land \text{all-numberps}\,(y))$

THEOREM: all-numberps-nodes-implies-all-numberps-parent
$\text{all-numberps}\,(\text{nodes-rec}\,(\mathit{flag},\ \mathit{tree}))$
$\rightarrow \quad \text{all-numberps}\,(\text{parent-rec}\,(\mathit{flag},\ \mathit{child},\ \mathit{tree}))$

THEOREM: all-numberps-nodes-implies-all-numberps-car-parent
$\text{all-numberps}\,(\text{nodes-rec}\,(\mathit{flag},\ \mathit{tree}))$
$\rightarrow \quad (\mathrm{car}\,(\text{parent-rec}\,(\mathit{flag},\ \mathit{child},\ \mathit{tree})) \in \mathbf{N})$

THEOREM: parent-not-litatom
$\text{all-numberps}\,(\text{nodes-rec}\,(\mathit{flag},\ \mathit{tree}))$
$\rightarrow \quad ((\mathrm{pack}\,(x) = \mathrm{car}\,(\text{parent-rec}\,(\mathit{flag},\ \mathit{child},\ \mathit{tree}))) = \mathbf{f})$

THEOREM: all-numberps-forest-implies-all-numberps-roots
$\text{all-numberps}\,(\text{nodes-rec}\,(\texttt{'forest},\ \mathit{forest})) \rightarrow \text{all-numberps}\,(\mathrm{roots}\,(\mathit{forest}))$

THEOREM: all-numberps-nodes-implies-all-numberps-children
all-numberps (nodes-rec (*flag*, *tree*))
$\rightarrow$ all-numberps (children-rec (*flag*, *parent*, *tree*))

THEOREM: dl-preserves-instance-of-dl
(treep (*tree*)
$\wedge$ (*down-link* $\in$ down-links (nodes (*tree*), *tree*))
$\wedge$ n (*old*, *new*, *statement*)
$\wedge$ (*statement* $\in$ tree-prg (*tree*))
$\wedge$ dl (down-links (nodes (*tree*), *tree*), *old*))
$\rightarrow$ ((empty (*down-link*, *new*)
$\wedge$ (status (car (*down-link*), *new*) = status (cdr (*down-link*), *new*)))
$\vee$ ((channel (*down-link*, *new*) = list (`'find`))
$\wedge$ (status (car (*down-link*), *new*) = `'started`)
$\wedge$ (status (cdr (*down-link*), *new*) = `'not-started`)))

THEOREM: dl-preserves-sublist
(dl (down-links (nodes (*tree*), *tree*), *old*)
$\wedge$ treep (*tree*)
$\wedge$ n (*old*, *new*, *statement*)
$\wedge$ (*statement* $\in$ tree-prg (*tree*))
$\wedge$ sublistp (*sublist*, down-links (nodes (*tree*), *tree*)))
$\rightarrow$ dl (*sublist*, *new*)

THEOREM: dl-preserves-dl
(dl (down-links (nodes (*tree*), *tree*), *old*)
$\wedge$ treep (*tree*)
$\wedge$ n (*old*, *new*, *statement*)
$\wedge$ (*statement* $\in$ tree-prg (*tree*)))
$\rightarrow$ dl (down-links (nodes (*tree*), *tree*), *new*)

THEOREM: member-up-links
(*up-link* $\in$ up-links (*nodes*, *tree*))
= ((car (*up-link*) $\in$ *nodes*)
$\wedge$ (cdr (*up-link*) = parent (car (*up-link*), *tree*))
$\wedge$ listp (*up-link*))

THEOREM: zero-not-reported-implies-children-reported
((number-not-reported (*children*, *parent*, *state*) $\simeq$ `0`) $\wedge$ (*child* $\in$ *children*))
$\rightarrow$ ((cdr (assoc (cons (`'status`, *child*), *state*)) = `'started`)
$\wedge$ (outstanding (*child*, *state*) $\simeq$ `0`)
$\wedge$ ($\neg$ listp (cdr (assoc (cons (*child*, *parent*), *state*)))))

THEOREM: dl-ul-no-preserves-instance-of-ul
(treep (*tree*)

$\wedge \quad (\mathit{up\text{-}link} \in \text{up-links}\,(\text{cdr}\,(\text{nodes}\,(\mathit{tree})),\ \mathit{tree}))$
$\wedge \quad \text{n}\,(\mathit{old},\ \mathit{new},\ \mathit{statement})$
$\wedge \quad (\mathit{statement} \in \text{tree-prg}\,(\mathit{tree}))$
$\wedge \quad \text{dl}\,(\text{down-links}\,(\text{nodes}\,(\mathit{tree}),\ \mathit{tree}),\ \mathit{old})$
$\wedge \quad \text{ul}\,(\text{up-links}\,(\text{cdr}\,(\text{nodes}\,(\mathit{tree})),\ \mathit{tree}),\ \mathit{old})$
$\wedge \quad \text{no}\,(\text{nodes}\,(\mathit{tree}),\ \mathit{tree},\ \mathit{old}))$
$\rightarrow \quad (\text{empty}\,(\mathit{up\text{-}link},\ \mathit{new})$
$\qquad \vee \quad ((\text{channel}\,(\mathit{up\text{-}link},\ \mathit{new})$
$\qquad\qquad = \quad \text{list}\,(\text{found-value}\,(\text{car}\,(\mathit{up\text{-}link}),\ \mathit{new})))$
$\qquad\qquad \wedge \quad \text{done}\,(\text{car}\,(\mathit{up\text{-}link}),\ \mathit{new})))$

THEOREM: dl-ul-no-preserves-ul-sublist
$(\text{treep}\,(\mathit{tree})$
$\wedge \quad \text{n}\,(\mathit{old},\ \mathit{new},\ \mathit{statement})$
$\wedge \quad (\mathit{statement} \in \text{tree-prg}\,(\mathit{tree}))$
$\wedge \quad \text{dl}\,(\text{down-links}\,(\text{nodes}\,(\mathit{tree}),\ \mathit{tree}),\ \mathit{old})$
$\wedge \quad \text{ul}\,(\text{up-links}\,(\text{cdr}\,(\text{nodes}\,(\mathit{tree})),\ \mathit{tree}),\ \mathit{old})$
$\wedge \quad \text{no}\,(\text{nodes}\,(\mathit{tree}),\ \mathit{tree},\ \mathit{old})$
$\wedge \quad \text{sublistp}\,(\mathit{sublist},\ \text{up-links}\,(\text{cdr}\,(\text{nodes}\,(\mathit{tree})),\ \mathit{tree})))$
$\rightarrow \quad \text{ul}\,(\mathit{sublist},\ \mathit{new})$

THEOREM: dl-ul-no-preserves-ul
$(\text{treep}\,(\mathit{tree})$
$\wedge \quad \text{n}\,(\mathit{old},\ \mathit{new},\ \mathit{statement})$
$\wedge \quad (\mathit{statement} \in \text{tree-prg}\,(\mathit{tree}))$
$\wedge \quad \text{dl}\,(\text{down-links}\,(\text{nodes}\,(\mathit{tree}),\ \mathit{tree}),\ \mathit{old})$
$\wedge \quad \text{ul}\,(\text{up-links}\,(\text{cdr}\,(\text{nodes}\,(\mathit{tree})),\ \mathit{tree}),\ \mathit{old})$
$\wedge \quad \text{no}\,(\text{nodes}\,(\mathit{tree}),\ \mathit{tree},\ \mathit{old}))$
$\rightarrow \quad \text{ul}\,(\text{up-links}\,(\text{cdr}\,(\text{nodes}\,(\mathit{tree})),\ \mathit{tree}),\ \mathit{new})$

THEOREM: parent-not-started-implies-all-empty-and-not-started
$((\text{status}\,(\mathit{parent},\ \mathit{state}) = \texttt{'not-started})$
$\wedge \quad \text{dl}\,(\text{rfp}\,(\mathit{parent},\ \mathit{children}),\ \mathit{state}))$
$\rightarrow \quad (\text{all-empty}\,(\text{rfp}\,(\mathit{parent},\ \mathit{children}),\ \mathit{state})$
$\qquad \wedge \quad \text{not-started}\,(\mathit{children},\ \mathit{state}))$

THEOREM: start-preserves-no-for-parent
$((\mathit{parent} \in \mathbf{N})$
$\wedge \quad (\mathit{parent} \notin \mathit{children})$
$\wedge \quad \text{not-started}\,(\mathit{children},\ \mathit{old})$
$\wedge \quad \text{sublistp}\,(\text{rfp}\,(\mathit{parent},\ \mathit{children}),\ \text{rfp}\,(\mathit{parent},\ \mathit{excpt}))$
$\wedge \quad \text{changed}\,(\mathit{old},$
$\qquad\qquad \mathit{new},$
$\qquad\qquad \text{append}\,(\text{list}\,(\text{cons}\,(\texttt{'status},\ \mathit{parent}),$
$\qquad\qquad\qquad\qquad \text{cons}\,(\texttt{'found-value},\ \mathit{parent}),$

$$
\begin{array}{l}
\qquad\qquad\qquad\qquad\qquad \text{cons}\,(\texttt{'outstanding}, \ \mathit{parent})),\\
\qquad\qquad\qquad\qquad \text{rfp}\,(\mathit{parent}, \ \mathit{excpt}))))\\
\to \quad ((\text{number-not-reported}\,(\mathit{children}, \ \mathit{parent}, \ \mathit{new}) = \text{length}\,(\mathit{children}))\\
\quad\ \wedge \quad (\text{min-of-reported}\,(\mathit{children}, \ \mathit{parent}, \ \mathit{new}, \ \mathit{value}) = \mathit{value}))
\end{array}
$$

THEOREM: unchanged-preserves-no

$$
\begin{array}{l}
\text{changed}\,(\mathit{old}, \ \mathit{new}, \ \mathbf{nil})\\
\to \quad ((\text{number-not-reported}\,(\mathit{children}, \ \mathit{parent}, \ \mathit{new})\\
\qquad = \quad \text{number-not-reported}\,(\mathit{children}, \ \mathit{parent}, \ \mathit{old}))\\
\quad\ \wedge \quad (\text{min-of-reported}\,(\mathit{children}, \ \mathit{parent}, \ \mathit{new}, \ \mathit{value})\\
\qquad\quad = \quad \text{min-of-reported}\,(\mathit{children}, \ \mathit{parent}, \ \mathit{old}, \ \mathit{value})))
\end{array}
$$

THEOREM: start-preserves-no-for-rest-of-tree

$$
\begin{array}{l}
((\mathit{root} \in \mathbf{N})\\
\ \wedge \quad (\mathit{parent} \in \mathbf{N})\\
\ \wedge \quad (\mathit{parent} \notin \mathit{children})\\
\ \wedge \quad (\mathit{root} \notin \mathit{children})\\
\ \wedge \quad (\mathit{root} \neq \mathit{parent})\\
\ \wedge \quad \text{changed}\,(\mathit{old},\\
\qquad\qquad\quad \mathit{new},\\
\qquad\qquad\quad \text{append}\,(\text{list}\,(\text{cons}\,(\texttt{'status}, \ \mathit{root}),\\
\qquad\qquad\qquad\qquad\qquad \text{cons}\,(\texttt{'found-value}, \ \mathit{root}),\\
\qquad\qquad\qquad\qquad\qquad \text{cons}\,(\texttt{'outstanding}, \ \mathit{root})),\\
\qquad\qquad\qquad\qquad \text{rfp}\,(\mathit{root}, \ \mathit{excpt}))))\\
\to \quad ((\text{number-not-reported}\,(\mathit{children}, \ \mathit{parent}, \ \mathit{new})\\
\qquad = \quad \text{number-not-reported}\,(\mathit{children}, \ \mathit{parent}, \ \mathit{old}))\\
\quad\ \wedge \quad (\text{min-of-reported}\,(\mathit{children}, \ \mathit{parent}, \ \mathit{new}, \ \mathit{value})\\
\qquad\quad = \quad \text{min-of-reported}\,(\mathit{children}, \ \mathit{parent}, \ \mathit{old}, \ \mathit{value})))
\end{array}
$$

THEOREM: length-rfp

$$
\text{length}\,(\text{rfp}\,(\mathit{parent}, \ \mathit{children})) = \text{length}\,(\mathit{children})
$$

THEOREM: start-preserves-instance-of-no

$$
\begin{array}{l}
(\text{treep}\,(\mathit{tree})\\
\ \wedge \quad \text{start}\,(\mathit{old}, \ \mathit{new}, \ \text{car}\,(\mathit{tree}), \ \text{rfp}\,(\text{car}\,(\mathit{tree}), \ \text{children}\,(\text{car}\,(\mathit{tree}), \ \mathit{tree})))\\
\ \wedge \quad (\mathit{node} \in \text{nodes}\,(\mathit{tree}))\\
\ \wedge \quad \text{dl}\,(\text{rfp}\,(\text{car}\,(\mathit{tree}), \ \text{children}\,(\text{car}\,(\mathit{tree}), \ \mathit{tree})), \ \mathit{old})\\
\ \wedge \quad (\text{status}\,(\mathit{node}, \ \mathit{new}) = \texttt{'started})\\
\ \wedge \quad ((\text{status}\,(\mathit{node}, \ \mathit{old}) = \texttt{'started})\\
\qquad \to \quad ((\text{outstanding}\,(\mathit{node}, \ \mathit{old})\\
\qquad\qquad = \quad \text{number-not-reported}\,(\text{children}\,(\mathit{node}, \ \mathit{tree}), \ \mathit{node}, \ \mathit{old}))\\
\qquad\quad\ \wedge \quad (\text{found-value}\,(\mathit{node}, \ \mathit{old})\\
\qquad\qquad\quad = \quad \text{min-of-reported}\,(\text{children}\,(\mathit{node}, \ \mathit{tree}),\\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{node},\\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{old},
\end{array}
$$

node-value (*node*, *old*))))))

$\rightarrow$ ((outstanding (*node*, *new*)

$=$ number-not-reported (children (*node*, *tree*), *node*, *new*))

$\wedge$ (found-value (*node*, *new*)

$=$ min-of-reported (children (*node*, *tree*),
node,
new,
node-value (*node*, *new*))))

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
(($parent \in \mathbf{N}$)
$\wedge$ ($child \in \mathbf{N}$)
$\wedge$ ($parent \neq child$)
$\wedge$ all-numberps (*children*)
$\wedge$ setp (*children*)
$\wedge$ ($parent \notin children$)
$\wedge$ (channel (cons (*child*, *parent*), *old*) $=$ list (found-value (*child*, *old*)))
$\wedge$ done (*child*, *old*)
$\wedge$ (channel (cons (*child*, *parent*), *new*) $=$ receive (cons (*child*, *parent*), *old*))
$\wedge$ changed (*old*,
new,
list (cons (*child*, *parent*),
cons (`'outstanding`, *parent*),
cons (`'found-value`, *parent*))))
$\rightarrow$ (min-of-reported (*children*, *parent*, *new*, *value*)
$=$ **if** $child \in children$
then min (found-value (*child*, *old*),
min-of-reported (*children*, *parent*, *old*, *value*))
else min-of-reported (*children*, *parent*, *old*, *value*) **endif**)

THEOREM: min-of-reported-of-non-root
(($root \in \mathbf{N}$)

$$
\begin{array}{ll}
\wedge & (child \in \mathbf{N}) \\
\wedge & (parent \in \mathbf{N}) \\
\wedge & \text{all-numberps}\,(children) \\
\wedge & \text{setp}\,(children) \\
\wedge & (parent \notin children) \\
\wedge & (root \neq parent) \\
\wedge & (root \notin children) \\
\wedge & \text{changed}\,(old, \\
& \qquad\qquad new, \\
& \qquad\qquad \text{list}\,(\text{cons}\,(child,\ root), \\
& \qquad\qquad\qquad \text{cons}\,(\texttt{'outstanding},\ root), \\
& \qquad\qquad\qquad \text{cons}\,(\texttt{'found-value},\ root)))) \\
\to & (\text{min-of-reported}\,(children,\ parent,\ new,\ value) \\
& \ = \ \text{min-of-reported}\,(children,\ parent,\ old,\ value))
\end{array}
$$

THEOREM: number-not-reported-of-non-root

$$
\begin{array}{ll}
\multicolumn{2}{l}{((root \in \mathbf{N})} \\
\wedge & (child \in \mathbf{N}) \\
\wedge & (parent \in \mathbf{N}) \\
\wedge & \text{all-numberps}\,(children) \\
\wedge & \text{setp}\,(children) \\
\wedge & (parent \notin children) \\
\wedge & (root \neq parent) \\
\wedge & (root \notin children) \\
\wedge & \text{changed}\,(old, \\
& \qquad\qquad new, \\
& \qquad\qquad \text{list}\,(\text{cons}\,(child,\ root), \\
& \qquad\qquad\qquad \text{cons}\,(\texttt{'outstanding},\ root), \\
& \qquad\qquad\qquad \text{cons}\,(\texttt{'found-value},\ root)))) \\
\to & (\text{number-not-reported}\,(children,\ parent,\ new) \\
& \ = \ \text{number-not-reported}\,(children,\ parent,\ old))
\end{array}
$$

THEOREM: number-not-reported-of-root

$$
\begin{array}{ll}
\multicolumn{2}{l}{((parent \in \mathbf{N})} \\
\wedge & (child \in \mathbf{N}) \\
\wedge & (parent \neq child) \\
\wedge & \text{all-numberps}\,(children) \\
\wedge & \text{setp}\,(children) \\
\wedge & (parent \notin children) \\
\wedge & (\text{channel}\,(\text{cons}\,(child,\ parent),\ old) = \text{list}\,(\text{found-value}\,(child,\ old))) \\
\wedge & \text{done}\,(child,\ old) \\
\wedge & (\text{channel}\,(\text{cons}\,(child,\ parent),\ new) = \text{receive}\,(\text{cons}\,(child,\ parent),\ old)) \\
\wedge & \text{changed}\,(old, \\
& \qquad\qquad new,
\end{array}
$$

$$
\begin{array}{l}
\qquad\qquad\text{list}\,(\text{cons}\,(\mathit{child},\ \mathit{parent}),\\
\qquad\qquad\qquad\text{cons}\,(\texttt{'outstanding},\ \mathit{parent}),\\
\qquad\qquad\qquad\text{cons}\,(\texttt{'found-value},\ \mathit{parent}))))\\
\rightarrow\ \ (\text{number-not-reported}\,(\mathit{children},\ \mathit{parent},\ \mathit{new})\\
\qquad =\ \ \textbf{if}\ \mathit{child} \in \mathit{children}\\
\qquad\quad\ \ \textbf{then}\ \text{number-not-reported}\,(\mathit{children},\ \mathit{parent},\ \mathit{old}) - 1\\
\qquad\quad\ \ \textbf{else}\ \text{number-not-reported}\,(\mathit{children},\ \mathit{parent},\ \mathit{old})\ \textbf{endif})
\end{array}
$$

Theorem: setp-nodes-implies-setp-roots

$$
\begin{array}{l}
(\text{proper-tree}\,(\texttt{'forest},\ \mathit{forest}) \wedge \text{setp}\,(\text{nodes-rec}\,(\texttt{'forest},\ \mathit{forest})))\\
\rightarrow\ \ \text{setp}\,(\text{roots}\,(\mathit{forest}))
\end{array}
$$

Theorem: setp-nodes-setp-children

$$
\begin{array}{l}
(\text{proper-tree}\,(\mathit{flag},\ \mathit{tree}) \wedge \text{setp}\,(\text{nodes-rec}\,(\mathit{flag},\ \mathit{tree})))\\
\rightarrow\ \ \text{setp}\,(\text{children-rec}\,(\mathit{flag},\ \mathit{parent},\ \mathit{tree}))
\end{array}
$$

Theorem: root-receive-report-preserves-instance-of-no

$$
\begin{array}{l}
(\text{treep}\,(\mathit{tree})\\
\ \wedge\ \ (\mathit{child} \in \text{children}\,(\text{car}\,(\mathit{tree}),\ \mathit{tree}))\\
\ \wedge\ \ \text{root-receive-report}\,(\mathit{old},\ \mathit{new},\ \text{car}\,(\mathit{tree}),\ \text{cons}\,(\mathit{child},\ \text{car}\,(\mathit{tree})))\\
\ \wedge\ \ \text{ul}\,(\text{up-links}\,(\text{cdr}\,(\text{nodes}\,(\mathit{tree})),\ \mathit{tree}),\ \mathit{old})\\
\ \wedge\ \ (\mathit{node} \in \text{nodes}\,(\mathit{tree}))\\
\ \wedge\ \ (\text{status}\,(\mathit{node},\ \mathit{new}) = \texttt{'started})\\
\ \wedge\ \ ((\text{status}\,(\mathit{node},\ \mathit{old}) = \texttt{'started})\\
\qquad \rightarrow\ \ ((\text{outstanding}\,(\mathit{node},\ \mathit{old})\\
\qquad\qquad =\ \ \text{number-not-reported}\,(\text{children}\,(\mathit{node},\ \mathit{tree}),\ \mathit{node},\ \mathit{old}))\\
\qquad\quad\ \wedge\ \ (\text{found-value}\,(\mathit{node},\ \mathit{old})\\
\qquad\qquad\quad =\ \ \text{min-of-reported}\,(\text{children}\,(\mathit{node},\ \mathit{tree}),\\
\qquad\qquad\qquad\qquad\qquad\qquad \mathit{node},\\
\qquad\qquad\qquad\qquad\qquad\qquad \mathit{old},\\
\qquad\qquad\qquad\qquad\qquad\qquad \text{node-value}\,(\mathit{node},\ \mathit{old}))))))\\
\rightarrow\ \ ((\text{outstanding}\,(\mathit{node},\ \mathit{new})\\
\qquad =\ \ \text{number-not-reported}\,(\text{children}\,(\mathit{node},\ \mathit{tree}),\ \mathit{node},\ \mathit{new}))\\
\quad\ \wedge\ \ (\text{found-value}\,(\mathit{node},\ \mathit{new})\\
\qquad\quad =\ \ \text{min-of-reported}\,(\text{children}\,(\mathit{node},\ \mathit{tree}),\\
\qquad\qquad\qquad\qquad\qquad \mathit{node},\\
\qquad\qquad\qquad\qquad\qquad \mathit{new},\\
\qquad\qquad\qquad\qquad\qquad \text{node-value}\,(\mathit{node},\ \mathit{new}))))
\end{array}
$$

Theorem: receive-find-preserves-no-for-rest-of-tree

$$
\begin{array}{l}
((\mathit{node} \in \mathbf{N})\\
\ \wedge\ \ (\mathit{parent\text{-}of\text{-}node} \in \mathbf{N})\\
\ \wedge\ \ (\mathit{parent} \in \mathbf{N})\\
\ \wedge\ \ (\mathit{parent} \neq \mathit{node})\\
\ \wedge\ \ (\mathit{node} \notin \mathit{children})
\end{array}
$$

$$
\begin{array}{l}
\quad \wedge \quad \text{changed}\,(\mathit{old}, \\
\qquad\qquad\qquad \mathit{new}, \\
\qquad\qquad\qquad \text{append}\,(\text{list}\,(\text{cons}\,(\mathit{parent\text{-}of\text{-}node},\ \mathit{node}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{cons}\,(\mathit{node},\ \mathit{parent\text{-}of\text{-}node}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{cons}\,(\texttt{'status},\ \mathit{node}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{cons}\,(\texttt{'found-value},\ \mathit{node}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{cons}\,(\texttt{'outstanding},\ \mathit{node})), \\
\qquad\qquad\qquad\qquad\qquad \text{rfp}\,(\mathit{node},\ \mathit{excpt})))) \\
\rightarrow \quad ((\text{number-not-reported}\,(\mathit{children},\ \mathit{parent},\ \mathit{new}) \\
\qquad\quad = \quad \text{number-not-reported}\,(\mathit{children},\ \mathit{parent},\ \mathit{old})) \\
\qquad \wedge \quad (\text{min-of-reported}\,(\mathit{children},\ \mathit{parent},\ \mathit{new},\ \mathit{value}) \\
\qquad\qquad = \quad \text{min-of-reported}\,(\mathit{children},\ \mathit{parent},\ \mathit{old},\ \mathit{value})))
\end{array}
$$

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

$$
\begin{array}{l}
((\mathit{node} \in \mathbf{N}) \\
\quad \wedge \quad (\mathit{parent\text{-}of\text{-}node} \in \mathbf{N}) \\
\quad \wedge \quad (\mathit{node} \notin \mathit{children}) \\
\quad \wedge \quad \text{not-started}\,(\mathit{children},\ \mathit{old}) \\
\quad \wedge \quad \text{sublistp}\,(\text{rfp}\,(\mathit{node},\ \mathit{children}),\ \text{rfp}\,(\mathit{node},\ \mathit{excpt})) \\
\quad \wedge \quad \text{changed}\,(\mathit{old}, \\
\qquad\qquad\qquad \mathit{new}, \\
\qquad\qquad\qquad \text{append}\,(\text{list}\,(\text{cons}\,(\mathit{parent\text{-}of\text{-}node},\ \mathit{node}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{cons}\,(\mathit{node},\ \mathit{parent\text{-}of\text{-}node}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{cons}\,(\texttt{'status},\ \mathit{node}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{cons}\,(\texttt{'found-value},\ \mathit{node}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{cons}\,(\texttt{'outstanding},\ \mathit{node})), \\
\qquad\qquad\qquad\qquad\qquad \text{rfp}\,(\mathit{node},\ \mathit{excpt})))) \\
\rightarrow \quad ((\text{number-not-reported}\,(\mathit{children},\ \mathit{node},\ \mathit{new}) = \text{length}\,(\mathit{children})) \\
\qquad \wedge \quad (\text{min-of-reported}\,(\mathit{children},\ \mathit{node},\ \mathit{new},\ \mathit{value}) \\
\qquad\qquad = \quad \text{min-of-reported}\,(\mathit{children},\ \mathit{node},\ \mathit{old},\ \mathit{value})))
\end{array}
$$

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

$$
\begin{array}{l}
((\mathit{node} \in \mathbf{N}) \\
\quad \wedge \quad (\mathit{parent\text{-}of\text{-}node} \in \mathbf{N}) \\
\quad \wedge \quad (\mathit{node} \neq \mathit{parent\text{-}of\text{-}node}) \\
\quad \wedge \quad (\text{status}\,(\mathit{node},\ \mathit{old}) \neq \texttt{'started}) \\
\quad \wedge \quad ((\text{outstanding}\,(\mathit{node},\ \mathit{new}) \simeq \texttt{0}) \\
\qquad \rightarrow \quad (\neg\ \text{empty}\,(\text{cons}\,(\mathit{node},\ \mathit{parent\text{-}of\text{-}node}),\ \mathit{new}))) \\
\quad \wedge \quad \text{changed}\,(\mathit{old}, \\
\qquad\qquad\qquad \mathit{new}, \\
\qquad\qquad\qquad \text{append}\,(\text{list}\,(\text{cons}\,(\mathit{parent\text{-}of\text{-}node},\ \mathit{node}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{cons}\,(\mathit{node},\ \mathit{parent\text{-}of\text{-}node}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{cons}\,(\texttt{'status},\ \mathit{node}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{cons}\,(\texttt{'found-value},\ \mathit{node}),
\end{array}
$$

$\text{cons}\,(\texttt{'outstanding},\ \textit{node})),$
$\text{rfp}\,(\textit{node},\ \textit{excpt}))))$
$\rightarrow\quad ((\text{number-not-reported}\,(\textit{children},\ \textit{parent-of-node},\ \textit{new})$
$=\quad \text{number-not-reported}\,(\textit{children},\ \textit{parent-of-node},\ \textit{old}))$
$\land\quad (\text{min-of-reported}\,(\textit{children},\ \textit{parent-of-node},\ \textit{new},\ \textit{value})$
$=\quad \text{min-of-reported}\,(\textit{children},\ \textit{parent-of-node},\ \textit{old},\ \textit{value})))$

THEOREM: dl-of-append
$\text{dl}\,(\text{append}\,(a,\ b),\ \textit{state}) = (\text{dl}\,(a,\ \textit{state}) \land \text{dl}\,(b,\ \textit{state}))$

THEOREM: down-links-1-rfp
$\text{down-links-1}\,(\textit{parent},\ \textit{children}) = \text{rfp}\,(\textit{parent},\ \textit{children})$

THEOREM: dl-down-links-implies-dl-rfp
$(\text{dl}\,(\text{down-links}\,(\textit{nodes},\ \textit{tree}),\ \textit{state}) \land (\textit{node} \in \textit{nodes}))$
$\rightarrow\quad \text{dl}\,(\text{rfp}\,(\textit{node},\ \text{children}\,(\textit{node},\ \textit{tree})),\ \textit{state})$

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
$(\text{treep}\,(\textit{tree})$
$\land\quad (\textit{node} \in \text{cdr}\,(\text{nodes}\,(\textit{tree})))$
$\land\quad \text{receive-find}\,(\textit{old},$
$\textit{new},$
$\textit{node},$
$\text{cons}\,(\text{parent}\,(\textit{node},\ \textit{tree}),\ \textit{node}),$
$\text{cons}\,(\textit{node},\ \text{parent}\,(\textit{node},\ \textit{tree})),$
$\text{rfp}\,(\textit{node},\ \text{children}\,(\textit{node},\ \textit{tree})))$
$\land\quad \text{dl}\,(\text{down-links}\,(\text{nodes}\,(\textit{tree}),\ \textit{tree}),\ \textit{old})$
$\land\quad (n \in \text{nodes}\,(\textit{tree}))$
$\land\quad (\text{status}\,(n,\ \textit{new}) = \texttt{'started})$
$\land\quad ((\text{status}\,(n,\ \textit{old}) = \texttt{'started})$
$\rightarrow\quad ((\text{outstanding}\,(n,\ \textit{old})$
$=\quad \text{number-not-reported}\,(\text{children}\,(n,\ \textit{tree}),\ n,\ \textit{old}))$
$\land\quad (\text{found-value}\,(n,\ \textit{old})$
$=\quad \text{min-of-reported}\,(\text{children}\,(n,\ \textit{tree}),$
$n,$
$\textit{old},$
$\text{node-value}\,(n,\ \textit{old}))))))$

$$
\begin{array}{l}
\to \quad ((\text{outstanding}\,(n,\ \mathit{new}) = \text{number-not-reported}\,(\text{children}\,(n,\ \mathit{tree}),\ n,\ \mathit{new})) \\
\qquad \land \quad (\text{found-value}\,(n,\ \mathit{new}) \\
\qquad\qquad = \quad \text{min-of-reported}\,(\text{children}\,(n,\ \mathit{tree}), \\
\qquad\qquad\qquad\qquad n, \\
\qquad\qquad\qquad\qquad \mathit{new}, \\
\qquad\qquad\qquad\qquad \text{node-value}\,(n,\ \mathit{new}))))
\end{array}
$$

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

$$
\begin{array}{l}
((\mathit{node} \in \mathbf{N}) \\
\land \quad (\mathit{parent\text{-}of\text{-}node} \in \mathbf{N}) \\
\land \quad (\mathit{child\text{-}of\text{-}node} \in \mathbf{N}) \\
\land \quad (\mathit{parent} \in \mathbf{N}) \\
\land \quad (\mathit{parent} \neq \mathit{node}) \\
\land \quad (\mathit{node} \notin \mathit{children}) \\
\land \quad \text{changed}\,(\mathit{old}, \\
\qquad\qquad \mathit{new}, \\
\qquad\qquad \text{list}\,(\text{cons}\,(\mathit{child\text{-}of\text{-}node},\ \mathit{node}), \\
\qquad\qquad\quad \text{cons}\,(\mathit{node},\ \mathit{parent\text{-}of\text{-}node}), \\
\qquad\qquad\quad \text{cons}\,(\texttt{'outstanding},\ \mathit{node}), \\
\qquad\qquad\quad \text{cons}\,(\texttt{'found-value},\ \mathit{node})))) \\
\to \quad ((\text{number-not-reported}\,(\mathit{children},\ \mathit{parent},\ \mathit{new}) \\
\qquad = \quad \text{number-not-reported}\,(\mathit{children},\ \mathit{parent},\ \mathit{old})) \\
\qquad \land \quad (\text{min-of-reported}\,(\mathit{children},\ \mathit{parent},\ \mathit{new},\ \mathit{value}) \\
\qquad\qquad = \quad \text{min-of-reported}\,(\mathit{children},\ \mathit{parent},\ \mathit{old},\ \mathit{value})))
\end{array}
$$

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

$$
\begin{array}{l}
((\mathit{node} \in \mathbf{N}) \\
\land \quad (\mathit{parent} \in \mathbf{N}) \\
\land \quad (\mathit{child} \in \mathbf{N}) \\
\land \quad (\mathit{node} \notin \mathit{children}) \\
\land \quad \text{all-numberps}\,(\mathit{children}) \\
\land \quad \text{setp}\,(\mathit{children}) \\
\land \quad (\text{channel}\,(\text{cons}\,(\mathit{child},\ \mathit{node}),\ \mathit{old}) = \text{list}\,(\text{found-value}\,(\mathit{child},\ \mathit{old}))) \\
\land \quad \text{done}\,(\mathit{child},\ \mathit{old}) \\
\land \quad (\text{channel}\,(\text{cons}\,(\mathit{child},\ \mathit{node}),\ \mathit{new}) = \text{receive}\,(\text{cons}\,(\mathit{child},\ \mathit{node}),\ \mathit{old})) \\
\land \quad \text{changed}\,(\mathit{old}, \\
\qquad\qquad \mathit{new}, \\
\qquad\qquad \text{list}\,(\text{cons}\,(\mathit{child},\ \mathit{node}), \\
\qquad\qquad\quad \text{cons}\,(\mathit{node},\ \mathit{parent}), \\
\qquad\qquad\quad \text{cons}\,(\texttt{'outstanding},\ \mathit{node}), \\
\qquad\qquad\quad \text{cons}\,(\texttt{'found-value},\ \mathit{node})))) \\
\to \quad ((\text{number-not-reported}\,(\mathit{children},\ \mathit{node},\ \mathit{new}) \\
\qquad = \quad \textbf{if } \mathit{child} \in \mathit{children} \\
\qquad\qquad \textbf{then } \text{number-not-reported}\,(\mathit{children},\ \mathit{node},\ \mathit{old}) - 1
\end{array}
$$

$\quad\quad\quad$ **else** number-not-reported ($children$, $node$, old) **endif**)
$\wedge$ (min-of-reported ($children$, $node$, new, $value$)
$\quad$ = **if** $child \in children$
$\quad\quad$ **then** min (found-value ($child$, old),
$\quad\quad\quad\quad$ min-of-reported ($children$, $node$, old, $value$))
$\quad\quad$ **else** min-of-reported ($children$, $node$, old, $value$) **endif**))

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

Theorem: child-member-cdr-nodes
(proper-tree (`'tree`, $tree$)
$\wedge$ setp (nodes-rec (`'tree`, $tree$))
$\wedge$ ($child \in$ children-rec (`'tree`, $node$, $tree$)))
$\rightarrow$ ($child \in$ cdr (nodes-rec (`'tree`, $tree$)))

Theorem: receive-report-preserves-instance-of-no
(treep ($tree$)
$\wedge$ ($node \in$ cdr (nodes ($tree$)))
$\wedge$ ($child \in$ children ($node$, $tree$))
$\wedge$ ($n \in$ nodes ($tree$))
$\wedge$ receive-report (old,
$\quad\quad$ new,
$\quad\quad$ $node$,
$\quad\quad$ cons ($child$, $node$),
$\quad\quad$ cons ($node$, parent ($node$, $tree$)))
$\wedge$ (status (n, new) = `'started`)
$\wedge$ ul (up-links (cdr (nodes ($tree$)), $tree$), old)
$\wedge$ no (nodes ($tree$), $tree$, old)
$\wedge$ dl (down-links (nodes ($tree$), $tree$), old))
$\rightarrow$ ((outstanding (n, new) = number-not-reported (children (n, $tree$), n, new))

$\wedge$ (found-value (n, new)
= min-of-reported (children (n, tree),
n,
new,
node-value (n, new))))

THEOREM: dl-ul-no-preserves-instance-of-no
(treep (tree)
$\wedge$ n $(\mathit{old}, \mathit{new}, \mathit{statement})$
$\wedge$ $(\mathit{statement} \in$ tree-prg $(\mathit{tree}))$
$\wedge$ dl (down-links (nodes (tree), tree), old)
$\wedge$ ul (up-links (cdr (nodes (tree)), tree), old)
$\wedge$ no (nodes (tree), tree, old)
$\wedge$ $(\mathit{node} \in$ nodes $(\mathit{tree}))$
$\wedge$ (status $(\mathit{node}, \mathit{new})$ = `'started`))
$\rightarrow$ ((outstanding $(\mathit{node}, \mathit{new})$
= number-not-reported (children $(\mathit{node}, \mathit{tree})$, node, new))
$\wedge$ (found-value $(\mathit{node}, \mathit{new})$
= min-of-reported (children $(\mathit{node}, \mathit{tree})$,
node,
new,
node-value $(\mathit{node}, \mathit{new})$)))

THEOREM: dl-ul-no-preserves-no-sublist
(treep (tree)
$\wedge$ n $(\mathit{old}, \mathit{new}, \mathit{statement})$
$\wedge$ $(\mathit{statement} \in$ tree-prg $(\mathit{tree}))$
$\wedge$ dl (down-links (nodes (tree), tree), old)
$\wedge$ ul (up-links (cdr (nodes (tree)), tree), old)
$\wedge$ no (nodes (tree), tree, old)
$\wedge$ sublistp $(\mathit{sublist}$, nodes (tree)))
$\rightarrow$ no $(\mathit{sublist}, \mathit{tree}, \mathit{new})$

THEOREM: inv-preserves-inv
(treep (tree)
$\wedge$ n $(\mathit{old}, \mathit{new}, \mathit{statement})$
$\wedge$ $(\mathit{statement} \in$ tree-prg $(\mathit{tree}))$
$\wedge$ inv $(\mathit{tree}, \mathit{old})$)
$\rightarrow$ inv $(\mathit{tree}, \mathit{new})$

THEOREM: inv-is-invariant
(initial-condition (`` `(and``
```
      (all-empty ',(all-channels tree) state)
      (not-started ',(nodes tree) state)),
```
tree-prg (tree))

$\wedge$ treep (*tree*))
$\rightarrow$ invariant (`` `(inv ',tree state) ``, tree-prg (*tree*))

THEOREM: outstanding-non-increasing
(treep (*tree*)
$\wedge$ (*statement* $\in$ tree-prg (*tree*))
$\wedge$ n (*old*, *new*, *statement*)
$\wedge$ dl (down-links (nodes (*tree*), *tree*), *old*)
$\wedge$ (*node* $\in$ nodes (*tree*)))
$\rightarrow$ (**if** status (*node*, *old*) = `'started` **then** outstanding (*node*, *old*)
else 1 + length (children (*node*, *tree*)) **endif**
$\not<$ **if** status (*node*, *new*) = `'started`
then outstanding (*node*, *new*)
else 1 + length (children (*node*, *tree*)) **endif**)

THEOREM: total-outstanding-non-increasing-sublist
(treep (*tree*)
$\wedge$ (*statement* $\in$ tree-prg (*tree*))
$\wedge$ n (*old*, *new*, *statement*)
$\wedge$ dl (down-links (nodes (*tree*), *tree*), *old*)
$\wedge$ sublistp (*sublist*, nodes (*tree*)))
$\rightarrow$ (total-outstanding (*sublist*, *tree*, *old*)
$\not<$ total-outstanding (*sublist*, *tree*, *new*))

THEOREM: total-outstanding-non-increasing
(treep (*tree*)
$\wedge$ (*statement* $\in$ tree-prg (*tree*))
$\wedge$ n (*old*, *new*, *statement*)
$\wedge$ dl (down-links (nodes (*tree*), *tree*), *old*))
$\rightarrow$ (total-outstanding (nodes (*tree*), *tree*, *old*)
$\not<$ total-outstanding (nodes (*tree*), *tree*, *new*))

THEOREM: position-append
position (append (a, b), e)
= **if** $e \in a$ **then** position (a, e)
else length (a) + position (b, e) **endif**

THEOREM: parents-position-decreases
((*node* $\in$ nodes-rec (*flag*, *tree*))
$\wedge$ setp (nodes-rec (*flag*, *tree*))
$\wedge$ proper-tree (*flag*, *tree*)
$\wedge$ **if** *flag* = `'tree` **then** car (*tree*) $\neq$ *node*
else *node* $\notin$ roots (*tree*) **endif**)
$\rightarrow$ (position (nodes-rec (*flag*, *tree*), car (parent-rec (*flag*, *node*, *tree*)))
$<$ position (nodes-rec (*flag*, *tree*), *node*))

Definition:
parent-to-root-induction $(node, tree)$
$= \textbf{if } (node \in \text{nodes}(tree))$
$\quad \wedge \quad \text{setp}(\text{nodes}(tree))$
$\quad \wedge \quad \text{proper-tree}(\texttt{'tree}, tree)$
$\quad \textbf{then if } \text{car}(tree) = node \textbf{ then t}$
$\qquad \textbf{else } \text{parent-to-root-induction}(\text{parent}(node, tree), tree) \textbf{ endif}$
$\quad \textbf{else t endif}$

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

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

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

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

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

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

Theorem: later-positions-are-in-suffix
$(\text{setp}(l)$
$\wedge \quad \text{suffix}(s, l)$
$\wedge \quad (x \in s)$

$\wedge \quad (y \in l)$
$\wedge \quad (\text{position}(l, x) < \text{position}(l, y)))$
$\rightarrow \quad (y \in s)$

DEFINITION:
all-done (*nodes*, *state*)
= **if** listp (*nodes*)
then if done (car (*nodes*), *state*) **then** all-done (cdr (*nodes*), *state*)
else f endif
else t endif

THEOREM: all-done-implies-done
(all-done (*nodes*, *state*) $\wedge$ (*node* $\in$ *nodes*)) $\rightarrow$ done (*node*, *state*)

THEOREM: all-done-implies-all-done-sublist
(all-done (*nodes*, *state*) $\wedge$ sublistp (*sublist*, *nodes*))
$\rightarrow$ all-done (*sublist*, *state*)

DEFINITION:
ulnks (*children*, *parent*)
= **if** listp (*children*)
then cons (cons (car (*children*), *parent*), ulnks (cdr (*children*), *parent*))
else nil endif

THEOREM: all-done-and-all-empty-implies-number-not-reported-0
(all-done (*children*, *state*) $\wedge$ all-empty (ulnks (*children*, *parent*), *state*))
$\rightarrow$ (number-not-reported (*children*, *parent*, *state*) = `0`)

THEOREM: all-empty-implies-all-empty-sublist
(all-empty (*channels*, *state*) $\wedge$ sublistp (*sublist*, *channels*))
$\rightarrow$ all-empty (*sublist*, *state*)

THEOREM: sublist-ulnks
(proper-tree (`'tree`, *tree*)
$\wedge$ sublistp (*sublist*, children (*parent*, *tree*))
$\wedge$ setp (nodes (*tree*)))
$\rightarrow$ sublistp (ulnks (*sublist*, *parent*), up-links (cdr (nodes (*tree*)), *tree*))

THEOREM: child-of-node-in-suffix-is-in-suffix
(proper-tree (`'tree`, *tree*)
$\wedge$ setp (nodes (*tree*))
$\wedge$ (*child* $\in$ children (*node*, *tree*))
$\wedge$ suffix (*nodes*, nodes (*tree*))
$\wedge$ (*node* $\in$ *nodes*))
$\rightarrow$ (*child* $\in$ cdr (*nodes*))

THEOREM: children-are-suffix-of-sublist-generalized
(proper-tree ($\texttt{'tree}$, *tree*)
 $\land$ setp (nodes (*tree*))
 $\land$ suffix (*nodes*, nodes (*tree*))
 $\land$ (*node* $\in$ *nodes*)
 $\land$ sublistp (*sublist*, children-rec ($\texttt{'tree}$, *node*, *tree*)))
$\rightarrow$ sublistp (*sublist*, cdr (*nodes*))

THEOREM: all-nodes-are-done
(proper-tree ($\texttt{'tree}$, *tree*)
 $\land$ setp (nodes (*tree*))
 $\land$ all-empty (down-links (nodes (*tree*), *tree*), *state*)
 $\land$ all-empty (up-links (cdr (nodes (*tree*)), *tree*), *state*)
 $\land$ dl (down-links (nodes (*tree*), *tree*), *state*)
 $\land$ ul (up-links (cdr (nodes (*tree*)), *tree*), *state*)
 $\land$ no (nodes (*tree*), *tree*, *state*)
 $\land$ (status (car (*tree*), *state*) = $\texttt{'started}$)
 $\land$ suffix (*nodes*, nodes (*tree*)))
$\rightarrow$ all-done (*nodes*, *state*)

THEOREM: all-done-implies-total-outstanding-0
all-done (*nodes*, *state*) $\rightarrow$ (total-outstanding (*nodes*, *tree*, *state*) = $\texttt{0}$)

THEOREM: all-empty-root-started-implies-total-outstanding-0
(proper-tree ($\texttt{'tree}$, *tree*)
 $\land$ setp (nodes (*tree*))
 $\land$ inv (*tree*, *state*)
 $\land$ all-empty (down-links (nodes (*tree*), *tree*), *state*)
 $\land$ all-empty (up-links (cdr (nodes (*tree*)), *tree*), *state*)
 $\land$ (status (car (*tree*), *state*) = $\texttt{'started}$))
$\rightarrow$ (total-outstanding (nodes (*tree*), *tree*, *state*) = $\texttt{0}$)

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
(($\neg$ all-empty (*channels*, *state*)) $\land$ (**f** $\notin$ *channels*))
$\rightarrow$ (listp (cdr (assoc (full-channel (*channels*, *state*), *state*)))
  $\land$ (full-channel (*channels*, *state*) $\in$ *channels*)
  $\land$ full-channel (*channels*, *state*))

THEOREM: not-total-outstanding-0-implies-full-channel

$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{proper-tree}\,(\texttt{'tree},\ \mathit{tree})}\\
\wedge & \text{setp}\,(\text{nodes}\,(\mathit{tree}))\\
\wedge & \text{inv}\,(\mathit{tree},\ \mathit{state})\\
\wedge & ((\text{status}\,(\text{car}\,(\mathit{tree}),\ \mathit{state}) = \texttt{'started})\\
 & \quad \vee \quad (\text{status}\,(\text{car}\,(\mathit{tree}),\ \mathit{state}) = \texttt{'not-started}))\\
\wedge & (\text{total-outstanding}\,(\text{nodes}\,(\mathit{tree}),\ \mathit{tree},\ \mathit{state}) \neq \texttt{0}))\\
\rightarrow & ((\text{status}\,(\text{car}\,(\mathit{tree}),\ \mathit{state}) = \texttt{'not-started})\\
 & \quad \vee \quad \text{full-channel}\,(\text{down-links}\,(\text{nodes}\,(\mathit{tree}),\ \mathit{tree}),\ \mathit{state})\\
 & \quad \vee \quad \text{full-channel}\,(\text{up-links}\,(\text{cdr}\,(\text{nodes}\,(\mathit{tree})),\ \mathit{tree}),\ \mathit{state}))
\end{array}
$$

THEOREM: status-root-becomes-started-or-unchanged

$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{treep}\,(\mathit{tree}) \wedge (\mathit{statement} \in \text{tree-prg}\,(\mathit{tree})) \wedge \text{n}\,(\mathit{old},\ \mathit{new},\ \mathit{statement}))}\\
\rightarrow & ((\text{status}\,(\text{car}\,(\mathit{tree}),\ \mathit{new}) = \texttt{'started})\\
 & \quad \vee \quad (\text{status}\,(\text{car}\,(\mathit{tree}),\ \mathit{new}) = \text{status}\,(\text{car}\,(\mathit{tree}),\ \mathit{old})))
\end{array}
$$

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))
 ∧   treep (tree))
→   invariant (`(or
                  (equal
                   (status ',(car tree) state)
                   'started)
                  (equal
                   (status ',(car tree) state)
                   'not-started)),
               tree-prg (tree))
```

THEOREM: total-outstanding-decreases-sublist

$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{treep}\,(\mathit{tree})}\\
\wedge & \text{dl}\,(\text{down-links}\,(\text{nodes}\,(\mathit{tree}),\ \mathit{tree}),\ \mathit{old})\\
\wedge & \text{n}\,(\mathit{old},\ \mathit{new},\ \mathit{statement})\\
\wedge & (\mathit{statement} \in \text{tree-prg}\,(\mathit{tree}))\\
\wedge & \text{sublistp}\,(\mathit{nodes},\ \text{nodes}\,(\mathit{tree}))\\
\wedge & (\mathit{node} \in \mathit{nodes})\\
\wedge & (\textbf{if}\ \text{status}\,(\mathit{node},\ \mathit{new}) = \texttt{'started}\ \textbf{then}\ \text{outstanding}\,(\mathit{node},\ \mathit{new})\\
 & \quad \textbf{else}\ 1 + \text{length}\,(\text{children}\,(\mathit{node},\ \mathit{tree}))\ \textbf{endif}\\
 & \quad < \quad \textbf{if}\ \text{status}\,(\mathit{node},\ \mathit{old}) = \texttt{'started}\\
 & \qquad\quad \textbf{then}\ \text{outstanding}\,(\mathit{node},\ \mathit{old})\\
 & \qquad\quad \textbf{else}\ 1 + \text{length}\,(\text{children}\,(\mathit{node},\ \mathit{tree}))\ \textbf{endif}))\\
\rightarrow & (\text{total-outstanding}\,(\mathit{nodes},\ \mathit{tree},\ \mathit{new})\\
 & \quad < \quad \text{total-outstanding}\,(\mathit{nodes},\ \mathit{tree},\ \mathit{old}))
\end{array}
$$

DEFINITION:
tou $(old,\ new,\ node,\ tree)$
$$= \ (\textbf{if } \text{status}(node,\ new) = \texttt{'started} \textbf{ then } \text{outstanding}(node,\ new)$$
$$\textbf{else } 1 + \text{length}(\text{children}(node,\ tree)) \textbf{ endif}$$
$$< \ \textbf{if } \text{status}(node,\ old) = \texttt{'started}$$
$$\textbf{then } \text{outstanding}(node,\ old)$$
$$\textbf{else } 1 + \text{length}(\text{children}(node,\ tree)) \textbf{ endif})$$

THEOREM: total-outstanding-decreases
$$\begin{array}{ll}
& (\text{treep}(tree) \\
\wedge & \text{dl}(\text{down-links}(\text{nodes}(tree),\ tree),\ old) \\
\wedge & \text{n}(old,\ new,\ statement) \\
\wedge & (statement \in \text{tree-prg}(tree)) \\
\wedge & (node \in \text{nodes}(tree)) \\
\wedge & \text{tou}(old,\ new,\ node,\ tree)) \\
\rightarrow & (\text{total-outstanding}(\text{nodes}(tree),\ tree,\ new) \\
& \quad < \ \text{total-outstanding}(\text{nodes}(tree),\ tree,\ old))
\end{array}$$

THEOREM: start-decreases-tou
$$\begin{array}{ll}
& (\text{treep}(tree) \\
\wedge & (\text{status}(\text{car}(tree),\ old) = \texttt{'not-started}) \\
\wedge & \text{n}(old, \\
& \quad new, \\
& \quad \text{list}(\texttt{'start},\ \text{car}(tree),\ \text{rfp}(\text{car}(tree),\ \text{children}(\text{car}(tree),\ tree))))) \\
\rightarrow & \text{tou}(old,\ new,\ \text{car}(tree),\ tree)
\end{array}$$

THEOREM: others-preserve-root-not-started
$$\begin{array}{ll}
& (\text{treep}(tree) \\
\wedge & \text{n}(old,\ new,\ statement) \\
\wedge & (statement \in \text{tree-prg}(tree)) \\
\wedge & (statement \neq \text{list}(\texttt{'start}, \\
& \qquad \text{car}(tree), \\
& \qquad \text{rfp}(\text{car}(tree),\ \text{children}(\text{car}(tree),\ tree))))) \\
\rightarrow & (\text{cdr}(\text{assoc}(\text{cons}(\texttt{'status},\ \text{car}(tree)),\ new)) = \text{status}(\text{car}(tree),\ old))
\end{array}$$

THEOREM: root-receive-report-decreases-tou
$$\begin{array}{ll}
& (\text{treep}(tree) \\
\wedge & \text{listp}(\text{channel}(\text{cons}(child,\ \text{car}(tree)),\ old)) \\
\wedge & (child \in \text{children}(\text{car}(tree),\ tree)) \\
\wedge & \text{n}(old, \\
& \quad new, \\
& \quad \text{list}(\texttt{'root-receive-report},\ \text{car}(tree),\ \text{cons}(child,\ \text{car}(tree)))) \\
\wedge & \text{inv}(tree,\ old)) \\
\rightarrow & \text{tou}(old,\ new,\ \text{car}(tree),\ tree)
\end{array}$$

THEOREM: others-preserve-up-to-root-full

$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{treep}\,(\mathit{tree})} \\
\wedge & \text{listp}\,(\text{channel}\,(\text{cons}\,(\mathit{child},\ \text{car}\,(\mathit{tree})),\ \mathit{old})) \\
\wedge & (\mathit{child} \in \text{children}\,(\text{car}\,(\mathit{tree}),\ \mathit{tree})) \\
\wedge & (\mathit{statement} \in \text{tree-prg}\,(\mathit{tree})) \\
\wedge & (\mathit{statement} \neq \text{list}\,(\texttt{'root-receive-report},\ \text{car}\,(\mathit{tree}),\ \text{cons}\,(\mathit{child},\ \text{car}\,(\mathit{tree})))) \\
\wedge & \text{n}\,(\mathit{old},\ \mathit{new},\ \mathit{statement})) \\
\rightarrow & \text{listp}\,(\text{cdr}\,(\text{assoc}\,(\text{cons}\,(\mathit{child},\ \text{car}\,(\mathit{tree})),\ \mathit{new})))
\end{array}
$$

THEOREM: receive-find-decreases-tou

$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{treep}\,(\mathit{tree})} \\
\wedge & \text{listp}\,(\text{channel}\,(\text{cons}\,(\text{parent}\,(\mathit{node},\ \mathit{tree}),\ \mathit{node}),\ \mathit{old})) \\
\wedge & (\mathit{node} \in \text{cdr}\,(\text{nodes}\,(\mathit{tree}))) \\
\wedge & \text{n}\,(\mathit{old},\ \mathit{new},\ \text{list}\,(\texttt{'receive-find},\ \mathit{node},\ \text{cons}\,(\text{parent}\,(\mathit{node},\ \mathit{tree}),\ \mathit{node}),\ \text{cons}\,(\mathit{node},\ \text{parent}\,(\mathit{node},\ \mathit{tree})),\ \text{rfp}\,(\mathit{node},\ \text{children}\,(\mathit{node},\ \mathit{tree})))) \\
\wedge & \text{inv}\,(\mathit{tree},\ \mathit{old})) \\
\rightarrow & \text{tou}\,(\mathit{old},\ \mathit{new},\ \mathit{node},\ \mathit{tree})
\end{array}
$$

THEOREM: others-preserve-down-to-node-full

$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{treep}\,(\mathit{tree})} \\
\wedge & \text{listp}\,(\text{channel}\,(\text{cons}\,(\text{parent}\,(\mathit{node},\ \mathit{tree}),\ \mathit{node}),\ \mathit{old})) \\
\wedge & (\mathit{node} \in \text{cdr}\,(\text{nodes}\,(\mathit{tree}))) \\
\wedge & (\mathit{statement} \in \text{tree-prg}\,(\mathit{tree})) \\
\wedge & (\mathit{statement} \neq \text{list}\,(\texttt{'receive-find},\ \mathit{node},\ \text{cons}\,(\text{parent}\,(\mathit{node},\ \mathit{tree}),\ \mathit{node}),\ \text{cons}\,(\mathit{node},\ \text{parent}\,(\mathit{node},\ \mathit{tree})),\ \text{rfp}\,(\mathit{node},\ \text{children}\,(\mathit{node},\ \mathit{tree})))) \\
\wedge & \text{n}\,(\mathit{old},\ \mathit{new},\ \mathit{statement})) \\
\rightarrow & \text{listp}\,(\text{cdr}\,(\text{assoc}\,(\text{cons}\,(\text{car}\,(\text{parent-rec}\,(\texttt{'tree},\ \mathit{node},\ \mathit{tree})),\ \mathit{node}),\ \mathit{new})))
\end{array}
$$

THEOREM: receive-report-decreases-tou

$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{treep}\,(\mathit{tree})} \\
\wedge & \text{listp}\,(\text{channel}\,(\text{cons}\,(\mathit{child},\ \mathit{node}),\ \mathit{old})) \\
\wedge & (\mathit{node} \in \text{cdr}\,(\text{nodes}\,(\mathit{tree}))) \\
\wedge & (\mathit{child} \in \text{children}\,(\mathit{node},\ \mathit{tree})) \\
\wedge & \text{n}\,(\mathit{old},\ \mathit{new},
\end{array}
$$

$\text{list}\,(\texttt{'receive-report},$
$\quad node,$
$\quad \text{cons}\,(child,\ node),$
$\quad \text{cons}\,(node,\ \text{parent}\,(node,\ tree))))$
$\wedge\ \ \text{inv}\,(tree,\ old))$
$\rightarrow\ \ \text{tou}\,(old,\ new,\ node,\ tree)$

THEOREM: others-preserve-up-to-node-full
$(\text{treep}\,(tree)$
$\wedge\ \ \text{listp}\,(\text{channel}\,(\text{cons}\,(child,\ node),\ old))$
$\wedge\ \ (node \in \text{cdr}\,(\text{nodes}\,(tree)))$
$\wedge\ \ (child \in \text{children}\,(node,\ tree))$
$\wedge\ \ (statement \neq \text{list}\,(\texttt{'receive-report},$
$\quad node,$
$\quad \text{cons}\,(child,\ node),$
$\quad \text{cons}\,(node,\ \text{parent}\,(node,\ tree))))$
$\wedge\ \ (statement \in \text{tree-prg}\,(tree))$
$\wedge\ \ \text{n}\,(old,\ new,\ statement))$
$\rightarrow\ \ \text{listp}\,(\text{cdr}\,(\text{assoc}\,(\text{cons}\,(child,\ node),\ new)))$

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 children-are-suffix-of-sublist-generalized.

EVENT: Disable child-of-node-in-suffix-is-in-suffix.

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 dl-and-all-empty-implies-root-defines-status.

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 root-receive-report-preserves-instance-of-no.

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 parent-not-started-implies-all-empty-and-not-started.

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

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

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

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-car-parent.

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-of-parent-not-node.

EVENT: Disable parent-not-grandchild.

EVENT: Disable parent-not-child.

EVENT: Disable member-down-links.

EVENT: Disable member-down-links-1.

EVENT: Disable ul-implies-instance-of-ul-not-empty-uplink.

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-implements-root-receive-report.

EVENT: Disable root-receive-report-func.

EVENT: Disable start-func-implements-start.

EVENT: Disable start-func.

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

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 proper-tree-tree-implies-nodes-exists.

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-down-links-in-nodes.

EVENT: Disable nodes-in-channels-append.

EVENT: Disable nodes-in-down-links-1-in-nodes.

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 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-not-in-children.

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 no-children-in-rest-of-tree.

EVENT: Disable no-children-in-rest-of-forest.

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-of-subtrees-in-complete-subtrees.

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

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

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

EVENT: Disable next-level-of-tree-in-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 parent-rec-children-rec.

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
$((\mathit{node} \in \mathrm{cdr}\,(\mathrm{nodes}\,(\mathit{tree}))) \wedge \mathrm{treep}\,(\mathit{tree})) \rightarrow (\mathit{node} \in \mathrm{nodes}\,(\mathit{tree}))$

THEOREM: total-outstanding-decreases-expanded
$(\mathrm{treep}\,(\mathit{tree})$
$\wedge \quad \mathrm{inv}\,(\mathit{tree},\ \mathit{old})$
$\wedge \quad \mathrm{n}\,(\mathit{old},\ \mathit{new},\ \mathit{statement})$
$\wedge \quad (\mathit{statement} \in \mathrm{tree\text{-}prg}\,(\mathit{tree}))$
$\wedge \quad (\mathit{node} \in \mathrm{nodes}\,(\mathit{tree}))$
$\wedge \quad \mathrm{tou}\,(\mathit{old},\ \mathit{new},\ \mathit{node},\ \mathit{tree}))$
$\rightarrow \quad (((\mathrm{total\text{-}outstanding}\,(\mathrm{nodes}\,(\mathit{tree}),\ \mathit{tree},\ \mathit{new})$
$\quad\quad < \quad \mathrm{total\text{-}outstanding}\,(\mathrm{nodes}\,(\mathit{tree}),\ \mathit{tree},\ \mathit{old}))$
$\quad\quad = \quad \mathbf{t})$
$\quad\quad \wedge \quad (\mathrm{total\text{-}outstanding}\,(\mathrm{nodes}\,(\mathit{tree}),\ \mathit{tree},\ \mathit{new})$

$< \quad$ total-outstanding (nodes (*tree*), *tree*, *old*)))

THEOREM: total-outstanding-decreases-expanded-count
(treep (*tree*)
 $\land$ inv (*tree*, *old*)
 $\land$ n (*old*, *new*, *statement*)
 $\land$ (*statement* $\in$ tree-prg (*tree*))
 $\land$ (*node* $\in$ nodes (*tree*))
 $\land$ tou (*old*, *new*, *node*, *tree*)
 $\land$ (total-outstanding (nodes (*tree*), *tree*, *old*) $=$ (1 + *count*)))
$\rightarrow$ (((total-outstanding (nodes (*tree*), *tree*, *new*) $<$ (1 + *count*)) $=$ **t**)
 $\land$ (total-outstanding (nodes (*tree*), *tree*, *new*) $<$ (1 + *count*)))

THEOREM: total-outstanding-non-increasing-expanded
(treep (*tree*)
 $\land$ (*statement* $\in$ tree-prg (*tree*))
 $\land$ n (*old*, *new*, *statement*)
 $\land$ inv (*tree*, *old*))
$\rightarrow$ (((total-outstanding (nodes (*tree*), *tree*, *old*)
 $<$ total-outstanding (nodes (*tree*), *tree*, *new*))
 $=$ **f**)
 $\land$ (total-outstanding (nodes (*tree*), *tree*, *old*)
 $\not<$ total-outstanding (nodes (*tree*), *tree*, *new*)))

THEOREM: total-outstanding-non-increasing-expanded-count
(treep (*tree*)
 $\land$ (*statement* $\in$ tree-prg (*tree*))
 $\land$ n (*old*, *new*, *statement*)
 $\land$ inv (*tree*, *old*)
 $\land$ (total-outstanding (nodes (*tree*), *tree*, *old*) $=$ (1 + *count*)))
$\rightarrow$ ((((1 + *count*) $<$ total-outstanding (nodes (*tree*), *tree*, *old*)) $=$ **f**)
 $\land$ ((1 + *count*) $\not<$ total-outstanding (nodes (*tree*), *tree*, *new*)))

THEOREM: key-statements-member-tree-prg
(treep (*tree*)
 $\rightarrow$ (list (`'start`, car (*tree*), rfp (car (*tree*), children (car (*tree*), *tree*)))
 $\in$ tree-prg (*tree*)))
$\land$ ((treep (*tree*) $\land$ (*child* $\in$ children (car (*tree*), *tree*)))
 $\rightarrow$ (list (`'root-receive-report`,
 car (*tree*),
 cons (*child*, car (*tree*)))
 $\in$ tree-prg (*tree*)))
$\land$ ((treep (*tree*) $\land$ (*node* $\in$ cdr (nodes (*tree*))))
 $\rightarrow$ (list (`'receive-find`,
 node,

cons (parent (*node*, *tree*), *node*),
cons (*node*, parent (*node*, *tree*)),
rfp (*node*, children (*node*, *tree*)))
$\in$ tree-prg (*tree*)))
$\wedge$ ((treep (*tree*)
$\wedge$ (*node* $\in$ cdr (nodes (*tree*)))
$\wedge$ (*child* $\in$ children (*node*, *tree*)))
$\rightarrow$ (list (`'receive-report`,
node,
cons (*child*, *node*),
cons (*node*, parent (*node*, *tree*)))
$\in$ tree-prg (*tree*)))

THEOREM: down-link-full-decreases-total-outstanding-ensures
(treep (*tree*)
$\wedge$ (*node* $\in$ cdr (nodes (*tree*)))
$\wedge$ inv (*tree*, *old*)
$\wedge$ listp (channel (cons (parent (*node*, *tree*), *node*), *old*))
$\wedge$ n (*old*,
new,
list (`'receive-find`,
node,
cons (parent (*node*, *tree*), *node*),
cons (*node*, parent (*node*, *tree*)),
rfp (*node*, children (*node*, *tree*)))))
$\rightarrow$ (total-outstanding (nodes (*tree*), *tree*, *new*)
$<$ total-outstanding (nodes (*tree*), *tree*, *old*))

THEOREM: down-link-full-unless
(treep (*tree*)
$\wedge$ (*node* $\in$ cdr (nodes (*tree*)))
$\wedge$ listp (channel (cons (parent (*node*, *tree*), *node*), *old*))
$\wedge$ (*statement* $\in$ tree-prg (*tree*))
$\wedge$ (*statement* $\neq$ list (`'receive-find`,
node,
cons (parent (*node*, *tree*), *node*),
cons (*node*, parent (*node*, *tree*)),
rfp (*node*, children (*node*, *tree*))))
$\wedge$ n (*old*, *new*, *statement*))
$\rightarrow$ listp (channel (cons (parent (*node*, *tree*), *node*), *new*))

THEOREM: down-link-full-decreases-total-outstanding
(treep (*tree*) $\wedge$ (*node* $\in$ cdr (nodes (*tree*))))
$\rightarrow$ 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

$\text{treep}(tree) \rightarrow (\text{car}(tree) \in \text{nodes}(tree))$

THEOREM: root-up-link-full-decreases-total-outstanding-ensures

$$
\begin{aligned}
&(\text{treep}(tree) \\
&\ \ \wedge\ \ \text{inv}(tree,\ old) \\
&\ \ \wedge\ \ (child \in \text{children}(\text{car}(tree),\ tree)) \\
&\ \ \wedge\ \ \text{listp}(\text{channel}(\text{cons}(child,\ \text{car}(tree)),\ old)) \\
&\ \ \wedge\ \ \text{n}(old, \\
&\qquad\quad new, \\
&\qquad\quad \text{list}(\texttt{'root-receive-report},\ \text{car}(tree),\ \text{cons}(child,\ \text{car}(tree))))) \\
&\rightarrow\ \ (\text{total-outstanding}(\text{nodes}(tree),\ tree,\ new) \\
&\qquad <\ \ \text{total-outstanding}(\text{nodes}(tree),\ tree,\ old))
\end{aligned}
$$

THEOREM: root-up-link-full-unless

$$
\begin{aligned}
&(\text{treep}(tree) \\
&\ \ \wedge\ \ (child \in \text{children}(\text{car}(tree),\ tree)) \\
&\ \ \wedge\ \ \text{listp}(\text{channel}(\text{cons}(child,\ \text{car}(tree)),\ old)) \\
&\ \ \wedge\ \ (statement \in \text{tree-prg}(tree)) \\
&\ \ \wedge\ \ (statement \neq \text{list}(\texttt{'root-receive-report}, \\
&\qquad\qquad\qquad\qquad \text{car}(tree), \\
&\qquad\qquad\qquad\qquad \text{cons}(child,\ \text{car}(tree)))) \\
&\ \ \wedge\ \ \text{n}(old,\ new,\ statement)) \\
&\rightarrow\ \ \text{listp}(\text{channel}(\text{cons}(child,\ \text{car}(tree)),\ new))
\end{aligned}
$$

THEOREM: up-link-full-decreases-total-outstanding-ensures

$$
\begin{array}{l}
(\text{treep}\,(\mathit{tree}) \\
\;\wedge\;\; \text{inv}\,(\mathit{tree},\ \mathit{old}) \\
\;\wedge\;\; (\mathit{node} \in \text{cdr}\,(\text{nodes}\,(\mathit{tree}))) \\
\;\wedge\;\; (\mathit{child} \in \text{children}\,(\mathit{node},\ \mathit{tree})) \\
\;\wedge\;\; \text{listp}\,(\text{channel}\,(\text{cons}\,(\mathit{child},\ \mathit{node}),\ \mathit{old})) \\
\;\wedge\;\; \text{n}\,(\mathit{old}, \\
\qquad\quad \mathit{new}, \\
\qquad\quad \text{list}\,(\texttt{'receive-report}, \\
\qquad\qquad\quad \mathit{node}, \\
\qquad\qquad\quad \text{cons}\,(\mathit{child},\ \mathit{node}), \\
\qquad\qquad\quad \text{cons}\,(\mathit{node},\ \text{parent}\,(\mathit{node},\ \mathit{tree}))))) \\
\rightarrow\;\; (\text{total-outstanding}\,(\text{nodes}\,(\mathit{tree}),\ \mathit{tree},\ \mathit{new}) \\
\qquad < \;\; \text{total-outstanding}\,(\text{nodes}\,(\mathit{tree}),\ \mathit{tree},\ \mathit{old}))
\end{array}
$$

THEOREM: up-link-full-unless

$$
\begin{array}{l}
(\text{treep}\,(\mathit{tree}) \\
\;\wedge\;\; (\mathit{node} \in \text{cdr}\,(\text{nodes}\,(\mathit{tree}))) \\
\;\wedge\;\; (\mathit{child} \in \text{children}\,(\mathit{node},\ \mathit{tree})) \\
\;\wedge\;\; \text{listp}\,(\text{channel}\,(\text{cons}\,(\mathit{child},\ \mathit{node}),\ \mathit{old})) \\
\;\wedge\;\; (\mathit{statement} \in \text{tree-prg}\,(\mathit{tree})) \\
\;\wedge\;\; (\mathit{statement} \neq \text{list}\,(\texttt{'receive-report}, \\
\qquad\qquad\qquad\qquad \mathit{node}, \\
\qquad\qquad\qquad\qquad \text{cons}\,(\mathit{child},\ \mathit{node}), \\
\qquad\qquad\qquad\qquad \text{cons}\,(\mathit{node},\ \text{parent}\,(\mathit{node},\ \mathit{tree})))) \\
\;\wedge\;\; \text{n}\,(\mathit{old},\ \mathit{new},\ \mathit{statement})) \\
\rightarrow\;\; \text{listp}\,(\text{channel}\,(\text{cons}\,(\mathit{child},\ \mathit{node}),\ \mathit{new}))
\end{array}
$$

THEOREM: member-cdr-nodes-equals

$$
\begin{array}{l}
(\text{treep}\,(\mathit{tree}) \wedge (\mathit{node} \neq \text{car}\,(\mathit{tree}))) \\
\rightarrow\;\; ((\mathit{node} \in \text{nodes}\,(\mathit{tree})) = (\mathit{node} \in \text{cdr}\,(\text{nodes}\,(\mathit{tree}))))
\end{array}
$$

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)))),
 `(lessp
```

```
          (total-outstanding
           ',(nodes tree)
           ',tree
           state)
          ',(add1 count)),
        tree-prg (tree))
```

THEOREM: not-started-root-decreases-total-outstanding-ensures

$$
\begin{array}{l}
(\mathrm{treep}\,(\mathit{tree}) \\
\ \land\ \ \mathrm{inv}\,(\mathit{tree},\ \mathit{old}) \\
\ \land\ \ (\mathrm{status}\,(\mathrm{car}\,(\mathit{tree}),\ \mathit{old}) = \texttt{'not-started}) \\
\ \land\ \ \mathrm{n}\,(\mathit{old}, \\
\qquad\quad \mathit{new}, \\
\qquad\quad \mathrm{list}\,(\texttt{'start},\ \mathrm{car}\,(\mathit{tree}),\ \mathrm{rfp}\,(\mathrm{car}\,(\mathit{tree}),\ \mathrm{children}\,(\mathrm{car}\,(\mathit{tree}),\ \mathit{tree}))))) \\
\to\ \ (\text{total-outstanding}\,(\mathrm{nodes}\,(\mathit{tree}),\ \mathit{tree},\ \mathit{new}) \\
\qquad < \ \ \text{total-outstanding}\,(\mathrm{nodes}\,(\mathit{tree}),\ \mathit{tree},\ \mathit{old}))
\end{array}
$$

THEOREM: not-started-root-unless

$$
\begin{array}{l}
(\mathrm{treep}\,(\mathit{tree}) \\
\ \land\ \ (\mathrm{status}\,(\mathrm{car}\,(\mathit{tree}),\ \mathit{old}) = \texttt{'not-started}) \\
\ \land\ \ (\mathit{statement} \in \text{tree-prg}\,(\mathit{tree})) \\
\ \land\ \ (\mathit{statement} \neq \mathrm{list}\,(\texttt{'start}, \\
\qquad\qquad\qquad\qquad\quad \mathrm{car}\,(\mathit{tree}), \\
\qquad\qquad\qquad\qquad\quad \mathrm{rfp}\,(\mathrm{car}\,(\mathit{tree}),\ \mathrm{children}\,(\mathrm{car}\,(\mathit{tree}),\ \mathit{tree})))) \\
\ \land\ \ \mathrm{n}\,(\mathit{old},\ \mathit{new},\ \mathit{statement})) \\
\to\ \ (\mathrm{status}\,(\mathrm{car}\,(\mathit{tree}),\ \mathit{new}) = \texttt{'not-started})
\end{array}
$$

THEOREM: not-started-root-decreases-total-outstanding

treep (*tree*)
→ leads-to (

```
`(and
  (inv ',tree state)
  (and
   (equal
    (status ',(car tree) state)
    'not-started)
   (equal
    (total-outstanding
     ',(nodes tree)
     ',tree
     state)
    ',(add1 count)))),
`(lessp
  (total-outstanding
   ',(nodes tree)
   ',tree
```

```
                  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

```
 (treep (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

```
 (treep (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)),
```

```
            `(equal
              (total-outstanding
               ',(nodes tree)
               ',tree
               state)
              0),
            tree-prg (tree))
```

THEOREM: termination

```
(treep (tree)
 ∧   initial-condition (`(and
                         (all-empty
                          ',(all-channels tree)
                          state)
                         (not-started ',(nodes tree) state)),
                       tree-prg (tree)))
→   leads-to ('(true),
            `(equal
              (total-outstanding
               ',(nodes tree)
               ',tree
               state)
              0),
            tree-prg (tree))
```

THEOREM: correctness-condition

```
(treep (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))
```

Index