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

`;;; Several Useful Theorems`

THEOREM: equal-iff
$((\text{truep}\,(a) \lor \text{falsep}\,(a)) \land (\text{truep}\,(b) \lor \text{falsep}\,(b)))$
$\rightarrow \quad ((a = b) = (a \leftrightarrow b))$

EVENT: Disable equal-iff.

DEFINITION:
length (list)
$=$ **if** listp (list) **then** $1 +$ length (cdr (list))
else 0 **endif**

DEFINITION:
nth $(\mathit{list},\ n)$
$=$ **if** $n \simeq 0$ **then** car (list)
else nth (cdr (list), $n - 1$) **endif**

DEFINITION:
position $(\mathit{prg},\ e)$
$=$ **if** listp (prg)
then if car $(\mathit{prg}) = e$ **then** 0
else $1 +$ position (cdr (prg), e) **endif**
else 0 **endif**

THEOREM: nth-member
$(n < \text{length}\,(\mathit{list})) \rightarrow (\text{nth}\,(\mathit{list},\ n) \in \mathit{list})$

THEOREM: listp-not-zero-length
$(\text{length}\,(\mathit{list}) = 0) = (\neg\ \text{listp}\,(\mathit{list}))$

THEOREM: position-zero
$(e \notin \mathit{prg}) \rightarrow (\text{position}\,(\mathit{prg},\ e) = \text{length}\,(\mathit{prg}))$

THEOREM: position-lessp
$(e \in \mathit{prg}) \rightarrow (\text{position}\,(\mathit{prg},\ e) < \text{length}\,(\mathit{prg}))$

THEOREM: nth-position
nth $(\mathit{prg}$, position $(\mathit{prg},\ e))$
$=$ **if** $e \in \mathit{prg}$ **then** e
else 0 **endif**

THEOREM: position-nth
$((e \in \mathit{list}) \wedge (\mathrm{nth}\,(\mathit{list},\ n) \neq e)) \rightarrow ((\mathrm{position}\,(\mathit{list},\ e) = n) = \mathbf{f})$

THEOREM: member-append
$(e \in \mathrm{append}\,(j,\ k)) = ((e \in j) \vee (e \in k))$

THEOREM: append-is-associative
$\mathrm{append}\,(\mathrm{append}\,(x,\ y),\ z) = \mathrm{append}\,(x,\ \mathrm{append}\,(y,\ z))$

```
;;; The Witness function for the Computation
```

DEFINITION: $\mathrm{mchoose}\,(\mathit{prg},\ i) = \mathrm{nth}\,(\mathit{prg},\ i\ \mathbf{mod}\ \mathrm{length}\,(\mathit{prg}))$

DEFINITION:
$$
\begin{aligned}
&\mathrm{mnext}\,(\mathit{prg},\ e,\ i)\\
= \quad & (i + \mathbf{if}\ \mathrm{position}\,(\mathit{prg},\ e) < (i\ \mathbf{mod}\ \mathrm{length}\,(\mathit{prg}))\\
&\qquad \mathbf{then}\ \mathrm{position}\,(\mathit{prg},\ e)\\
&\qquad\qquad + \quad (\mathrm{length}\,(\mathit{prg}) - (i\ \mathbf{mod}\ \mathrm{length}\,(\mathit{prg})))\\
&\qquad \mathbf{else}\ \mathrm{position}\,(\mathit{prg},\ e) - (i\ \mathbf{mod}\ \mathrm{length}\,(\mathit{prg}))\ \mathbf{endif})
\end{aligned}
$$

THEOREM: mnext-fixes
$((e \in \mathit{prg}) \wedge (i \notin \mathbf{N})) \rightarrow (\mathrm{mnext}\,(\mathit{prg},\ e,\ i) = \mathrm{mnext}\,(\mathit{prg},\ e,\ 0))$

THEOREM: numberp-mnext
$(e \in \mathit{prg}) \rightarrow (\mathrm{mnext}\,(\mathit{prg},\ e,\ i) \in \mathbf{N})$

THEOREM: mchoose-chooses
$\mathrm{listp}\,(\mathit{prg}) \rightarrow (\mathrm{mchoose}\,(\mathit{prg},\ i) \in \mathit{prg})$

THEOREM: mchoose-fixes
$(\mathrm{listp}\,(\mathit{prg}) \wedge (i \notin \mathbf{N})) \rightarrow (\mathrm{mchoose}\,(\mathit{prg},\ i) = \mathrm{mchoose}\,(\mathit{prg},\ 0))$

THEOREM: mnext-choice-1
$(e \in \mathit{prg}) \rightarrow (\mathrm{mnext}\,(\mathit{prg},\ e,\ i) \not< i)$

THEOREM: mnext-choice-2-simplified
$$
\begin{aligned}
&(e < n)\\
\rightarrow \quad & (((i + \mathbf{if}\ e < (i\ \mathbf{mod}\ n)\ \ \mathbf{then}\ e + (n - (i\ \mathbf{mod}\ n))\\
&\qquad\qquad \mathbf{else}\ e - (i\ \mathbf{mod}\ n)\ \mathbf{endif})\\
&\quad \mathbf{mod}\quad n)\\
&\quad = \quad \mathrm{fix}\,(e))
\end{aligned}
$$

THEOREM: mnext-choice-2
$(e \in \mathit{prg}) \rightarrow (\mathrm{mchoose}\,(\mathit{prg},\ \mathrm{mnext}\,(\mathit{prg},\ e,\ i)) = e)$

THEOREM: remainder-of-add1
$((x < (a \textbf{ mod } b)) \wedge (x < ((1 + a) \textbf{ mod } b)))$
$\rightarrow \quad (((1 + a) \textbf{ mod } b) = (1 + (a \textbf{ mod } b)))$

THEOREM: remainder-of-add1-1
$((x \not< (a \textbf{ mod } b)) \wedge (x < ((1 + a) \textbf{ mod } b))) \rightarrow ((a \textbf{ mod } b) = \text{fix}(x))$

THEOREM: remainder-of-add1-2
$((x < (a \textbf{ mod } b)) \wedge (x \not< ((1 + a) \textbf{ mod } b)))$
$\rightarrow \quad ((((1 + a) \textbf{ mod } b) = 0) \wedge ((a \textbf{ mod } b) = (b - 1)))$

THEOREM: remainder-of-add1-3
$((x \not< (a \textbf{ mod } b))$
$\wedge \quad (x < b)$
$\wedge \quad (x \in \mathbf{N})$
$\wedge \quad (x \neq (a \textbf{ mod } b))$
$\wedge \quad (x \not< ((1 + a) \textbf{ mod } b)))$
$\rightarrow \quad (((1 + a) \textbf{ mod } b) = (1 + (a \textbf{ mod } b)))$

THEOREM: remainder-of-add1-3-1
$((\text{position}(prg, e) \not< (i \textbf{ mod } \text{length}(prg)))$
$\wedge \quad (\text{position}(prg, e) \not< ((1 + i) \textbf{ mod } \text{length}(prg)))$
$\wedge \quad (\text{nth}(prg, i \textbf{ mod } \text{length}(prg)) \neq e)$
$\wedge \quad (e \in prg))$
$\rightarrow \quad (((1 + i) \textbf{ mod } \text{length}(prg)) = (1 + (i \textbf{ mod } \text{length}(prg))))$

EVENT: Disable mchoose.

EVENT: Disable mnext.

```
;;; The Statement Interpreter
```

DEFINITION:
$\text{n}(old, new, e) = \text{apply\$}(\text{car}(e), \text{append}(\text{list}(old, new), \text{cdr}(e)))$

DEFINITION: $\text{exists-successor}(old, e) \leftrightarrow \exists\, new\ \text{n}(old, new, e)$

THEOREM: exists-successor-implies
$\text{exists-successor}(old, e) \rightarrow \text{n}(old, \text{newx}(e, old), e)$

THEOREM: prove-exists-successor
$\text{n}(old, new, e) \rightarrow \text{exists-successor}(old, e)$

EVENT: Disable exists-successor.

DEFINITION:
ms $(prg,\ i)$
$=$ **if** $i \simeq 0$ **then nil**
elseif n (ms $(prg,\ i-1)$,
newx (mchoose $(prg,\ i-1)$, ms $(prg,\ i-1)$),
mchoose $(prg,\ i-1)$)
then newx (mchoose $(prg,\ i-1)$, ms $(prg,\ i-1)$)
else ms $(prg,\ i-1)$ **endif**

THEOREM: ms-transition-successful
(listp (prg) $\land$ n (ms $(prg,\ i)$, new, mchoose $(prg,\ i)$))
$\rightarrow$ n (ms $(prg,\ i)$, ms $(prg,\ 1+i)$, mchoose $(prg,\ i)$)

THEOREM: ms-transition-idle
(listp (prg)
$\land$ ($\neg$ n (ms $(prg,\ i)$, newx (mchoose $(prg,\ i)$, ms $(prg,\ i)$), mchoose $(prg,\ i)$)))
$\rightarrow$ (ms $(prg,\ 1+i)$ = ms $(prg,\ i)$)

```
;;; Characterizing an Arbitrary Computation
```

CONSERVATIVE AXIOM: computation
(listp (prg) $\rightarrow$ (choose $(prg,\ i) \in prg$))
$\land$ (($e \in prg$) $\rightarrow$ (next $(prg,\ e,\ i) \not< i$))
$\land$ (($e \in prg$) $\rightarrow$ (choose $(prg$, next $(prg,\ e,\ i)) = e$))
$\land$ ((listp (prg) $\land$ n (s $(prg,\ i)$, new, choose $(prg,\ i)$))
$\rightarrow$ n (s $(prg,\ i)$, s $(prg,\ 1+i)$, choose $(prg,\ i)$))
$\land$ ((listp (prg)
$\land$ ($\neg$ n (s $(prg,\ i)$, newx (choose $(prg,\ i)$, s $(prg,\ i)$), choose $(prg,\ i)$)))
$\rightarrow$ (s $(prg,\ 1+i)$ = s $(prg,\ i)$))
$\land$ (($e \in prg$) $\rightarrow$ (next $(prg,\ e,\ i) \in \mathbf{N}$))
$\land$ ((listp (prg) $\land$ $(i \notin \mathbf{N})$) $\rightarrow$ (s $(prg,\ i)$ = s $(prg,\ 0)$))
$\land$ ((listp (prg) $\land$ $(i \notin \mathbf{N})$) $\rightarrow$ (choose $(prg,\ i)$ = choose $(prg,\ 0)$))
$\land$ ((($e \in prg$) $\land$ $(i \notin \mathbf{N})$) $\rightarrow$ (next $(prg,\ e,\ i)$ = next $(prg,\ e,\ 0)$))

Simultaneously, we introduce the new function symbols *choose*, *next*, and *s*.
EVENT: Disable mnext-fixes.

EVENT: Disable numberp-mnext.

EVENT: Disable mchoose-chooses.

EVENT: Disable mchoose-fixes.

EVENT: Disable mnext-choice-1.

EVENT: Disable mnext-choice-2-simplified.

EVENT: Disable mnext-choice-2.

EVENT: Disable remainder-of-add1.

EVENT: Disable remainder-of-add1-1.

EVENT: Disable remainder-of-add1-2.

EVENT: Disable remainder-of-add1-3.

EVENT: Disable remainder-of-add1-3-1.

EVENT: Disable exists-successor-implies.

EVENT: Disable prove-exists-successor.

EVENT: Disable ms-transition-successful.

EVENT: Disable ms-transition-idle.

EVENT: Disable n.

`;;; The Predicate Interpreter`

DEFINITION:
eval $(pred,\ state)$ = eval\$ $(\mathtt{t},\ pred,$ list (cons $(\mathtt{'state},\ state)))$

THEOREM: eval-not
eval (list $(\mathtt{'not},\ p),\ state) = (\neg$ eval $(p,\ state))$

THEOREM: eval-and
eval (list $(\mathtt{'and},\ p,\ q),\ state) = ($eval $(p,\ state) \wedge$ eval $(q,\ state))$

THEOREM: eval-or
eval (list ('**or**, p, q), $state$) = (eval (p, $state$) $\vee$ eval (q, $state$))

THEOREM: eval-implies
eval (list ('**implies**, p, q), $state$) = (eval (p, $state$) $\rightarrow$ eval (q, $state$))

THEOREM: eval-iff
eval (list ('**iff**, p, q), $state$) = (eval (p, $state$) $\leftrightarrow$ eval (q, $state$))

THEOREM: eval-equal
eval (list ('**equal**, p, q), $state$) = (eval (p, $state$) = eval (q, $state$))

THEOREM: eval-true
eval ('**(true)**, $state$) = **t**

THEOREM: eval-false
eval ('**(false)**, $state$) = **f**

EVENT: Disable eval.

```
;;; The Stability Operator:  Unless
```

DEFINITION:
unless (p, q, prg)
$\leftrightarrow$ $\forall$ old, new, e ((($e \in prg$)
$\wedge$ (n (old, new, e)
$\wedge$ eval (list ('**and**, p, list ('**not**, q)), old)))
$\rightarrow$ eval (list ('**or**, p, q), new))

EVENT: Disable unless.

THEOREM: prove-unless
(((eu (p, prg, q) $\in prg$)
$\wedge$ n (oldu (p, prg, q), newu (p, prg, q), eu (p, prg, q))
$\wedge$ eval (p, oldu (p, prg, q))
$\wedge$ ($\neg$ eval (q, oldu (p, prg, q))))
$\rightarrow$ (eval (p, newu (p, prg, q)) $\vee$ eval (q, newu (p, prg, q))))
$\rightarrow$ unless (p, q, prg)

THEOREM: unless-implies
(unless (p, q, prg)
$\wedge$ ($e \in prg$)
$\wedge$ n (old, new, e)
$\wedge$ eval (p, old)
$\wedge$ ($\neg$ eval (q, old)))
$\rightarrow$ eval (list ('**or**, p, q), new)

EVENT: Disable prove-unless.

EVENT: Disable unless-implies.

```
;;; The Progress Operator:  Leads-To
```

DEFINITION:
leads-to $(p,\ q,\ prg)$
$\leftrightarrow \quad \forall\ i\ (\text{eval}\,(p,\ \text{s}\,(prg,\ i)) \rightarrow \exists\ j\ ((j\ \not< i) \wedge \text{eval}\,(q,\ \text{s}\,(prg,\ j))))$

EVENT: Disable leads-to.

THEOREM: prove-leads-to
$(\text{eval}\,(p,\ \text{s}\,(prg,\ \text{ileads}\,(p,\ prg,\ q)))$
$\quad \rightarrow \quad ((j\ \not< \text{ileads}\,(p,\ prg,\ q)) \wedge \text{eval}\,(q,\ \text{s}\,(prg,\ j))))$
$\rightarrow \quad \text{leads-to}\,(p,\ q,\ prg)$

THEOREM: leads-to-implies
$(\text{leads-to}\,(p,\ q,\ prg) \wedge \text{eval}\,(p,\ \text{s}\,(prg,\ i)))$
$\rightarrow \quad ((\text{jleads}\,(i,\ prg,\ q) \not< i) \wedge \text{eval}\,(q,\ \text{s}\,(prg,\ \text{jleads}\,(i,\ prg,\ q))))$

EVENT: Disable prove-leads-to.

EVENT: Disable leads-to-implies.

```
;;; The Most Effective Trace
```

DEFINITION:
$\text{schedulable}\,(prg) \leftrightarrow \forall\ i\ \exists\ new\ \text{n}\,(\text{s}\,(prg,\ i),\ new,\ \text{choose}\,(prg,\ i))$

EVENT: Disable schedulable.

THEOREM: prove-schedulable
$\text{n}\,(\text{s}\,(prg,\ \text{is}\,(prg)),\ new,\ \text{choose}\,(prg,\ \text{is}\,(prg))) \rightarrow \text{schedulable}\,(prg)$

THEOREM: schedulable-implies
$\text{schedulable}\,(prg) \rightarrow \text{n}\,(\text{s}\,(prg,\ i),\ \text{news}\,(i,\ prg),\ \text{choose}\,(prg,\ i))$

EVENT: Disable prove-schedulable.

EVENT: Disable schedulable-implies.

THEOREM: schedulable-implies-effective-computation
$(\text{schedulable}(prg) \land \text{listp}(prg))$
$\rightarrow \quad \text{n}(\text{s}(prg,\ i),\ \text{s}(prg,\ 1 + i),\ \text{choose}(prg,\ i))$

THEOREM: computation-n
$(\text{schedulable}(prg) \land (e \in prg))$
$\rightarrow \quad \text{n}(\text{s}(prg,\ \text{next}(prg,\ e,\ i)),\ \text{s}(prg,\ 1 + \text{next}(prg,\ e,\ i)),\ e)$

THEOREM: effective-idle
$\text{listp}(prg)$
$\rightarrow \quad ((\text{s}(prg,\ 1 + i) = \text{s}(prg,\ i))$
$\qquad \lor \quad \text{n}(\text{s}(prg,\ i),\ \text{s}(prg,\ 1 + i),\ \text{choose}(prg,\ i)))$

EVENT: Disable effective-idle.

```
;;; Leads-To Proof Rules
```

THEOREM: leads-to-transitive
$(\text{leads-to}(p,\ q,\ prg) \land \text{leads-to}(q,\ r,\ prg)) \rightarrow \text{leads-to}(p,\ r,\ prg)$

EVENT: Disable leads-to-transitive.

THEOREM: leads-to-transitive-general
$(\text{leads-to}(p\text{-}1,\ q\text{-}1,\ prg)$
$\land \quad \text{leads-to}(q\text{-}2,\ r\text{-}1,\ prg)$
$\land \quad (\text{eval}(q\text{-}1,\ \text{s}(prg,\ \text{jleads}(\text{ileads}(p,\ prg,\ r),\ prg,\ q\text{-}1)))$
$\qquad \rightarrow \quad \text{eval}(q\text{-}2,\ \text{s}(prg,\ \text{jleads}(\text{ileads}(p,\ prg,\ r),\ prg,\ q\text{-}1))))$
$\land \quad (\text{eval}(p,\ \text{s}(prg,\ \text{ileads}(p,\ prg,\ r))) \rightarrow \text{eval}(p\text{-}1,\ \text{s}(prg,\ \text{ileads}(p,\ prg,\ r))))$
$\land \quad (\text{eval}(r\text{-}1,\ \text{s}(prg,\ \text{jleads}(\text{jleads}(\text{ileads}(p,\ prg,\ r),\ prg,\ q\text{-}1),\ prg,\ r\text{-}1)))$
$\qquad \rightarrow \quad \text{eval}(r,$
$\qquad\qquad \text{s}(prg,\ \text{jleads}(\text{jleads}(\text{ileads}(p,\ prg,\ r),\ prg,\ q\text{-}1),\ prg,\ r\text{-}1)))))$
$\rightarrow \quad \text{leads-to}(p,\ r,\ prg)$

EVENT: Disable leads-to-transitive-general.

THEOREM: q-leads-to-q
$\text{leads-to}(q,\ q,\ prg)$

EVENT: Disable q-leads-to-q.

THEOREM: false-leads-to-anything
$\text{leads-to}(\texttt{'(false)},\ p,\ prg)$

EVENT: Disable false-leads-to-anything.

THEOREM: p-implies-q-leads-to
$(\text{eval}\,(p,\ \text{s}\,(prg,\ \text{ileads}\,(p,\ prg,\ q))) \rightarrow \text{eval}\,(q,\ \text{s}\,(prg,\ \text{ileads}\,(p,\ prg,\ q))))$
$\rightarrow\quad \text{leads-to}\,(p,\ q,\ prg)$

EVENT: Disable p-implies-q-leads-to.

THEOREM: leads-to-strengthen-left
$((\text{eval}\,(q,\ \text{s}\,(prg,\ \text{ileads}\,(q,\ prg,\ r))) \rightarrow \text{eval}\,(p,\ \text{s}\,(prg,\ \text{ileads}\,(q,\ prg,\ r))))$
$\land\quad \text{leads-to}\,(p,\ r,\ prg))$
$\rightarrow\quad \text{leads-to}\,(q,\ r,\ prg)$

EVENT: Disable leads-to-strengthen-left.

THEOREM: leads-to-weaken-right
$((\text{eval}\,(q,\ \text{s}\,(prg,\ \text{jleads}\,(\text{ileads}\,(p,\ prg,\ r),\ prg,\ q)))$
$\rightarrow\quad \text{eval}\,(r,\ \text{s}\,(prg,\ \text{jleads}\,(\text{ileads}\,(p,\ prg,\ r),\ prg,\ q))))$
$\land\quad \text{leads-to}\,(p,\ q,\ prg))$
$\rightarrow\quad \text{leads-to}\,(p,\ r,\ prg)$

EVENT: Disable leads-to-weaken-right.

THEOREM: leads-to-modify-both
$((\text{eval}\,(p,\ \text{s}\,(prg,\ \text{ileads}\,(p,\ prg,\ q))) \rightarrow \text{eval}\,(p\text{-}1,\ \text{s}\,(prg,\ \text{ileads}\,(p,\ prg,\ q))))$
$\land\quad (\text{eval}\,(q\text{-}1,\ \text{s}\,(prg,\ \text{jleads}\,(\text{ileads}\,(p,\ prg,\ q),\ prg,\ q\text{-}1)))$
$\rightarrow\quad \text{eval}\,(q,\ \text{s}\,(prg,\ \text{jleads}\,(\text{ileads}\,(p,\ prg,\ q),\ prg,\ q\text{-}1))))$
$\land\quad \text{leads-to}\,(p\text{-}1,\ q\text{-}1,\ prg))$
$\rightarrow\quad \text{leads-to}\,(p,\ q,\ prg)$

EVENT: Disable leads-to-modify-both.

THEOREM: disjoin-left
$(\text{leads-to}\,(p,\ r,\ prg) \land \text{leads-to}\,(q,\ r,\ prg))$
$\rightarrow\quad \text{leads-to}\,(\text{list}\,(\texttt{'or},\ p,\ q),\ r,\ prg)$

EVENT: Disable disjoin-left.

THEOREM: disjoin-left-general
$(\text{leads-to}\,(p\text{-}1,\ q\text{-}1,\ prg)$
$\land\quad \text{leads-to}\,(p\text{-}2,\ q\text{-}2,\ prg)$
$\land\quad (\text{eval}\,(p,\ \text{s}\,(prg,\ \text{ileads}\,(p,\ prg,\ q)))$

$\rightarrow$ eval (list ('or, *p-1*, *p-2*), s (*prg*, ileads (*p*, *prg*, *q*))))
$\wedge$ (eval (*q-1*, s (*prg*, jleads (ileads (*p*, *prg*, *q*), *prg*, *q-1*)))
$\rightarrow$ eval (*q*, s (*prg*, jleads (ileads (*p*, *prg*, *q*), *prg*, *q-1*))))
$\wedge$ (eval (*q-2*, s (*prg*, jleads (ileads (*p*, *prg*, *q*), *prg*, *q-2*)))
$\rightarrow$ eval (*q*, s (*prg*, jleads (ileads (*p*, *prg*, *q*), *prg*, *q-2*)))))
$\rightarrow$ leads-to (*p*, *q*, *prg*)

EVENT: Disable disjoin-left-general.

THEOREM: cancellation-leads-to
(leads-to (*p*, list ('or, *q*, *b*), *prg*) $\wedge$ leads-to (*b*, *r*, *prg*))
$\rightarrow$ leads-to (*p*, list ('or, *q*, *r*), *prg*)

EVENT: Disable cancellation-leads-to.

THEOREM: cancellation-leads-to-general
(leads-to (*p-1*, *d*, *prg*)
$\wedge$ leads-to (*b*, *r-1*, *prg*)
$\wedge$ (eval (list ('and, *d*, list ('not, *b*)),
s (*prg*, jleads (ileads (*p*, *prg*, *r*), *prg*, *d*)))
$\rightarrow$ eval (*r*, s (*prg*, jleads (ileads (*p*, *prg*, *r*), *prg*, *d*))))
$\wedge$ (eval (*r-1*, s (*prg*, jleads (jleads (ileads (*p*, *prg*, *r*), *prg*, *d*), *prg*, *r-1*)))
$\rightarrow$ eval (*r*, s (*prg*, jleads (jleads (ileads (*p*, *prg*, *r*), *prg*, *d*), *prg*, *r-1*))))
$\wedge$ (eval (*p*, s (*prg*, ileads (*p*, *prg*, *r*))) $\rightarrow$ eval (*p-1*, s (*prg*, ileads (*p*, *prg*, *r*)))))
$\rightarrow$ leads-to (*p*, *r*, *prg*)

EVENT: Disable cancellation-leads-to-general.

DEFINITION:
ensures-interval (*prg*, *q*, *top*, *i*)
= **if** $top < i$ **then** fix (*top*)
elseif eval (*q*, s (*prg*, *i*)) **then** fix (*i*)
else ensures-interval (*prg*, *q*, *top*, $1 + i$) **endif**

THEOREM: ensures-interval-fixes
(listp (*prg*) $\wedge$ ($i \notin \mathbf{N}$))
$\rightarrow$ (ensures-interval (*prg*, *q*, *top*, *i*) = ensures-interval (*prg*, *q*, *top*, 0))

THEOREM: ensures-interval-bigger
(ensures-interval (*prg*, *q*, *top*, *i*) $< i$) = ($top < i$)

THEOREM: psp-proves-something
(leads-to (*p*, *q*, *prg*)
$\wedge$ unless (*r*, *b*, *prg*)

$\wedge$ listp (prg)
$\wedge$ ($i \in \mathbf{N}$)
$\wedge$ eval (p, s (prg, $base$))
$\wedge$ eval (r, s (prg, i))
$\wedge$ (jleads ($base$, prg, q) $<$ top)
$\wedge$ (jleads ($base$, prg, q) $\not< i$)
$\wedge$ (final = list (`'or`, q, b)))
$\rightarrow$ eval (list (`'or`, list (`'and`, q, r), b),
s (prg, ensures-interval (prg, final, top, i)))

EVENT: Disable psp-proves-something.

THEOREM: psp
(leads-to (p, q, prg) $\wedge$ unless (r, b, prg) $\wedge$ listp (prg))
$\rightarrow$ leads-to (list (`'and`, p, r), list (`'or`, list (`'and`, q, r), b), prg)

EVENT: Disable psp.

THEOREM: psp-general
(leads-to (p, q, prg)
$\wedge$ unless (r, b, prg)
$\wedge$ (eval (pr, s (prg, ileads (pr, prg, qb)))
$\rightarrow$ eval (list (`'and`, p, r), s (prg, ileads (pr, prg, qb))))
$\wedge$ (eval (list (`'or`, list (`'and`, q, r), b),
s (prg,
jleads (ileads (pr, prg, qb), prg, list (`'or`, list (`'and`, q, r), b))))
$\rightarrow$ eval (qb,
s (prg,
jleads (ileads (pr, prg, qb),
prg,
list (`'or`, list (`'and`, q, r), b)))))
$\wedge$ listp (prg))
$\rightarrow$ leads-to (pr, qb, prg)

EVENT: Disable psp-general.

THEOREM: leads-to-true
leads-to (p, `'(true)`, prg)

`;;; Initial Condition and Invariants`

DEFINITION: initial-condition (ic, prg) = eval (ic, s (prg, 0))

EVENT: Disable initial-condition.

DEFINITION: invariant $(inv,\ prg) \leftrightarrow \forall\ i$ eval $(inv,\ \mathrm{s}\,(prg,\ i))$

EVENT: Disable invariant.

THEOREM: prove-invariant
eval $(inv,\ \mathrm{s}\,(prg,\ \mathrm{ii}\,(inv,\ prg))) \rightarrow$ invariant $(inv,\ prg)$

THEOREM: invariant-implies
invariant $(inv,\ prg) \rightarrow$ eval $(inv,\ \mathrm{s}\,(prg,\ i))$

EVENT: Disable prove-invariant.

THEOREM: invariant-consequence
(invariant $(p,\ prg)$
 $\wedge$ (eval $(p,\ \mathrm{s}\,(prg,\ \mathrm{ii}\,(q,\ prg))) \rightarrow$ eval $(q,\ \mathrm{s}\,(prg,\ \mathrm{ii}\,(q,\ prg)))))$
$\rightarrow$ invariant $(q,\ prg)$

THEOREM: invariants-persist-general
(unless $(p,$ `'(false)`$,\ prg) \wedge$ eval $(p,\ \mathrm{s}\,(prg,\ i)) \wedge$ listp $(prg) \wedge (j \not< i))$
$\rightarrow$ eval $(p,\ \mathrm{s}\,(prg,\ j))$

EVENT: Disable invariants-persist-general.

THEOREM: unless-proves-invariant
(initial-condition $(ic,\ prg)$
 $\wedge$ unless $(p,$ `'(false)`$,\ prg)$
 $\wedge$ (eval $(ic,\ \mathrm{s}\,(prg,\ 0)) \rightarrow$ eval $(p,\ \mathrm{s}\,(prg,\ 0)))$
 $\wedge$ listp $(prg))$
$\rightarrow$ invariant $(p,\ prg)$

THEOREM: leads-to-false-invariant
(leads-to $(p,$ `'(false)`$,\ prg)$
 $\wedge$ (eval (list (`'not`$,\ p),\ \mathrm{s}\,(prg,\ \mathrm{ii}\,(inv,\ prg)))$
  $\rightarrow$ eval $(inv,\ \mathrm{s}\,(prg,\ \mathrm{ii}\,(inv,\ prg)))))$
$\rightarrow$ invariant $(inv,\ prg)$

;;; Eventual Stability

DEFINITION:
eventually-stable $(r,\ prg) \leftrightarrow \exists\ i\ \forall\ j\ ((j \not< i) \rightarrow$ eval $(r,\ \mathrm{s}\,(prg,\ j)))$

EVENT: Disable eventually-stable.

THEOREM: prove-eventually-stable
$((\text{jes}\,(i,\ prg,\ r) \not< i) \rightarrow \text{eval}\,(r,\ \text{s}\,(prg,\ \text{jes}\,(i,\ prg,\ r))))$
$\rightarrow\quad \text{eventually-stable}\,(r,\ prg)$

THEOREM: eventually-stable-implies
$(\text{eventually-stable}\,(r,\ prg) \land (j \not< \text{ies}\,(prg,\ r))) \rightarrow \text{eval}\,(r,\ \text{s}\,(prg,\ j))$

EVENT: Disable prove-eventually-stable.

EVENT: Disable eventually-stable-implies.

THEOREM: not-leads-to-proves-eventually-stable
$((\neg\ \text{leads-to}\,(p,\ not\text{-}r,\ prg))$
$\land\quad ((\neg\ \text{eval}\,(not\text{-}r,\ \text{s}\,(prg,\ \text{jes}\,(\text{ileads}\,(p,\ prg,\ not\text{-}r),\ prg,\ r))))$
$\quad\rightarrow\quad \text{eval}\,(r,\ \text{s}\,(prg,\ \text{jes}\,(\text{ileads}\,(p,\ prg,\ not\text{-}r),\ prg,\ r)))))$
$\rightarrow\quad \text{eventually-stable}\,(r,\ prg)$

EVENT: Disable not-leads-to-proves-eventually-stable.

THEOREM: not-eventually-stable-proves-leads-to
$((\neg\ \text{eventually-stable}\,(not\text{-}q,\ prg))$
$\land\quad ((\neg\ \text{eval}\,(not\text{-}q,\ \text{s}\,(prg,\ \text{jes}\,(\text{ileads}\,(p,\ prg,\ q),\ prg,\ not\text{-}q))))$
$\quad\rightarrow\quad \text{eval}\,(q,\ \text{s}\,(prg,\ \text{jes}\,(\text{ileads}\,(p,\ prg,\ q),\ prg,\ not\text{-}q)))))$
$\rightarrow\quad \text{leads-to}\,(p,\ q,\ prg)$

EVENT: Disable not-eventually-stable-proves-leads-to.

THEOREM: true-leads-to-proves-not-eventually-stable
$(\text{leads-to}\,(\texttt{'(true)},\ not\text{-}r,\ prg)$
$\land\quad (\text{eval}\,(not\text{-}r,\ \text{s}\,(prg,\ \text{jleads}\,(\text{ies}\,(prg,\ r),\ prg,\ not\text{-}r)))$
$\quad\rightarrow\quad (\neg\ \text{eval}\,(r,\ \text{s}\,(prg,\ \text{jleads}\,(\text{ies}\,(prg,\ r),\ prg,\ not\text{-}r))))))$
$\rightarrow\quad (\neg\ \text{eventually-stable}\,(r,\ prg))$

EVENT: Disable true-leads-to-proves-not-eventually-stable.

THEOREM: eventually-stable-proves-not-true-leads-to
$(\text{eventually-stable}\,(not\text{-}q,\ prg)$
$\land\quad (\text{eval}\,(not\text{-}q,\ \text{s}\,(prg,\ \text{jleads}\,(\text{ies}\,(prg,\ not\text{-}q),\ prg,\ q)))$
$\quad\rightarrow\quad (\neg\ \text{eval}\,(q,\ \text{s}\,(prg,\ \text{jleads}\,(\text{ies}\,(prg,\ not\text{-}q),\ prg,\ q))))))$
$\rightarrow\quad (\neg\ \text{leads-to}\,(\texttt{'(true)},\ q,\ prg))$

EVENT: Disable eventually-stable-proves-not-true-leads-to.

THEOREM: eventually-stable-weaken

$$
\begin{array}{l}
(\text{eventually-stable}\,(p,\ prg) \\
\ \wedge\quad (\text{eval}\,(p,\ \text{s}\,(prg, \text{jes}\,(\text{ies}\,(prg,\ p),\ prg,\ r))) \\
\qquad\quad \to\quad \text{eval}\,(r,\ \text{s}\,(prg, \text{jes}\,(\text{ies}\,(prg,\ p),\ prg,\ r))))) \\
\to\quad \text{eventually-stable}\,(r,\ prg)
\end{array}
$$

EVENT: Disable eventually-stable-weaken.

THEOREM: eventually-stable-conjunction

$$
\begin{array}{l}
(\text{eventually-stable}\,(p,\ prg) \\
\ \wedge\quad \text{eventually-stable}\,(q,\ prg) \\
\ \wedge\quad (\text{eval}\,(\text{list}\,(\texttt{'and},\ p,\ q), \\
\qquad\qquad\quad \text{s}\,(prg, \\
\qquad\qquad\qquad \text{jes}\,(\textbf{if}\ \text{ies}\,(prg,\ p) < \text{ies}\,(prg,\ q)\ \ \textbf{then}\ \text{ies}\,(prg,\ q) \\
\qquad\qquad\qquad\qquad \textbf{else}\ \text{ies}\,(prg,\ p)\ \textbf{endif}, \\
\qquad\qquad\qquad\qquad prg, \\
\qquad\qquad\qquad\qquad r))) \\
\qquad\quad \to\quad \text{eval}\,(r, \\
\qquad\qquad\qquad\quad \text{s}\,(prg, \\
\qquad\qquad\qquad\qquad \text{jes}\,(\textbf{if}\ \text{ies}\,(prg,\ p) < \text{ies}\,(prg,\ q)\ \ \textbf{then}\ \text{ies}\,(prg,\ q) \\
\qquad\qquad\qquad\qquad\quad \textbf{else}\ \text{ies}\,(prg,\ p)\ \textbf{endif}, \\
\qquad\qquad\qquad\qquad\quad prg, \\
\qquad\qquad\qquad\qquad\quad r))))) \\
\to\quad \text{eventually-stable}\,(r,\ prg)
\end{array}
$$

EVENT: Disable eventually-stable-conjunction.

THEOREM: eventually-stable-false

$$
\begin{array}{l}
(\text{leads-to}\,(p,\ q,\ prg) \\
\ \wedge\quad (\text{eval}\,(q,\ \text{s}\,(prg, \text{jleads}\,(\text{ies}\,(prg,\ p),\ prg,\ q))) \\
\qquad\quad \to\quad (\neg\ \text{eval}\,(p,\ \text{s}\,(prg, \text{jleads}\,(\text{ies}\,(prg,\ p),\ prg,\ q)))))) \\
\to\quad (\neg\ \text{eventually-stable}\,(p,\ prg))
\end{array}
$$

EVENT: Disable eventually-stable-false.

THEOREM: stable-occurs-proves-eventually-stable

$$
\begin{array}{l}
(\text{listp}\,(prg) \wedge \text{unless}\,(p,\ \texttt{'(false)},\ prg) \wedge \text{leads-to}\,(\texttt{'(true)},\ p,\ prg)) \\
\to\quad \text{eventually-stable}\,(p,\ prg)
\end{array}
$$

EVENT: Disable stable-occurs-proves-eventually-stable.

`;;; The Basic Ensures Operator`

DEFINITION:
$$
\begin{array}{l}
\text{ensures}\,(p,\ q,\ prg) \\
\leftrightarrow \quad \exists\ e\ ((e \in prg) \\
\qquad\qquad \wedge \quad \forall\ old,\ new\ ((\text{n}\,(old,\ new,\ e) \\
\qquad\qquad\qquad\qquad\qquad\qquad \wedge \quad \text{eval}\,(\text{list}\,(\texttt{'and},\ p,\ \text{list}\,(\texttt{'not},\ q)),\ old)) \\
\qquad\qquad\qquad\qquad\qquad\qquad \rightarrow \quad \text{eval}\,(q,\ new)))
\end{array}
$$

EVENT: Disable ensures.

THEOREM: prove-ensures
$$
\begin{array}{l}
((e \in prg) \\
\ \wedge \quad ((\text{n}\,(\text{olde}\,(e,\ p,\ q),\ \text{newe}\,(e,\ p,\ q),\ e) \\
\qquad\quad \wedge \quad \text{eval}\,(p,\ \text{olde}\,(e,\ p,\ q)) \\
\qquad\quad \wedge \quad (\neg\ \text{eval}\,(q,\ \text{olde}\,(e,\ p,\ q)))) \\
\qquad\quad \rightarrow \quad \text{eval}\,(q,\ \text{newe}\,(e,\ p,\ q)))) \\
\rightarrow \quad \text{ensures}\,(p,\ q,\ prg)
\end{array}
$$

THEOREM: ensures-implies
$$
\begin{array}{l}
(\text{ensures}\,(p,\ q,\ prg) \rightarrow (\text{ee}\,(p,\ prg,\ q) \in prg)) \\
\wedge \quad ((\text{ensures}\,(p,\ q,\ prg) \\
\qquad\quad \wedge \quad \text{n}\,(old,\ new,\ \text{ee}\,(p,\ prg,\ q)) \\
\qquad\quad \wedge \quad \text{eval}\,(p,\ old) \\
\qquad\quad \wedge \quad (\neg\ \text{eval}\,(q,\ old))) \\
\qquad\quad \rightarrow \quad \text{eval}\,(q,\ new))
\end{array}
$$

EVENT: Disable prove-ensures.

EVENT: Disable ensures-implies.

THEOREM: ensures-proves-something
$$
\begin{array}{l}
(\text{ensures}\,(p,\ q,\ prg) \\
\ \wedge \quad \text{unless}\,(p,\ q,\ prg) \\
\ \wedge \quad \text{schedulable}\,(prg) \\
\ \wedge \quad (\text{next}\,(prg,\ \text{ee}\,(p,\ prg,\ q),\ base) < top) \\
\ \wedge \quad (\text{next}\,(prg,\ \text{ee}\,(p,\ prg,\ q),\ base) \not< i) \\
\ \wedge \quad (i \not< base) \\
\ \wedge \quad (i \in \mathbf{N}) \\
\ \wedge \quad \text{eval}\,(p,\ \text{s}\,(prg,\ i))) \\
\rightarrow \quad \text{eval}\,(q,\ \text{s}\,(prg,\ \text{ensures-interval}\,(prg,\ q,\ top,\ i)))
\end{array}
$$

THEOREM: the-interval-of-ensures
$(\text{ensures}(p,\ q,\ prg)$
$\ \wedge\quad \text{unless}(p,\ q,\ prg)$
$\ \wedge\quad \text{schedulable}(prg)$
$\ \wedge\quad \text{eval}(p,\ \text{s}(prg,\ i)))$
$\rightarrow\quad \text{eval}(q,\ \text{s}(prg,\ \text{ensures-interval}(prg,\ q,\ 1 + \text{next}(prg,\ \text{ee}(p,\ prg,\ q),\ i),\ i)))$

THEOREM: ensures-proves-leads-to
$(\text{schedulable}(prg) \wedge \text{unless}(p,\ q,\ prg) \wedge \text{ensures}(p,\ q,\ prg))$
$\rightarrow\quad \text{leads-to}(p,\ q,\ prg)$

EVENT: Disable ensures-proves-something.

EVENT: Disable the-interval-of-ensures.

EVENT: Disable ensures-proves-leads-to.

```
;;; Total Programs, Statements are Assignments
```

DEFINITION:
$\text{total}(prg) \leftrightarrow \forall\ e\ ((e \in prg) \rightarrow \forall\ old\ \exists\ new\ \text{n}(old,\ new,\ e))$

EVENT: Disable total.

THEOREM: prove-total
$((\text{et}(prg) \in prg) \rightarrow \text{n}(\text{oldt}(prg),\ new,\ \text{et}(prg))) \rightarrow \text{total}(prg)$

THEOREM: total-implies
$(\text{total}(prg) \wedge (e \in prg)) \rightarrow \text{n}(old,\ \text{newt}(e,\ old),\ e)$

EVENT: Disable prove-total.

EVENT: Disable total-implies.

THEOREM: total-implies-schedulable
$(\text{total}(prg) \wedge \text{listp}(prg)) \rightarrow \text{schedulable}(prg)$

```
;;; Enabled Transitions
```

DEFINITION:
enabling-condition $(c,\ e,\ prg)$
$\leftrightarrow \quad ((e \in prg)$
$\quad \wedge \quad \forall\ old,\ new\ (\mathrm{n}\,(old,\ new,\ e) \rightarrow \mathrm{eval}\,(c,\ old))$
$\quad \wedge \quad \forall\ old\ (\mathrm{eval}\,(c,\ old) \rightarrow \exists\ new\ \mathrm{n}\,(old,\ new,\ e)))$

EVENT: Disable enabling-condition.

THEOREM: prove-enabling-condition
$((e \in prg)$
$\wedge \quad (\mathrm{n}\,(\mathrm{oldc}\,(c,\ e),\ \mathrm{newc}\,(c,\ e),\ e) \rightarrow \mathrm{eval}\,(c,\ \mathrm{oldc}\,(c,\ e)))$
$\wedge \quad (\mathrm{eval}\,(c,\ \mathrm{oldc\text{-}1}\,(c,\ e)) \rightarrow \mathrm{n}\,(\mathrm{oldc\text{-}1}\,(c,\ e),\ new,\ e)))$
$\rightarrow \quad$ enabling-condition $(c,\ e,\ prg)$

THEOREM: enabling-condition-implies
(enabling-condition $(c,\ e,\ prg) \rightarrow (e \in prg))$
$\wedge \quad$ ((enabling-condition $(c,\ e,\ prg) \wedge \mathrm{n}\,(old,\ new,\ e)) \rightarrow \mathrm{eval}\,(c,\ old))$
$\wedge \quad$ ((enabling-condition $(c,\ e,\ prg) \wedge \mathrm{eval}\,(c,\ old))$
$\quad \rightarrow \quad \mathrm{n}\,(old,\ \mathrm{newc\text{-}1}\,(e,\ old),\ e))$

EVENT: Disable prove-enabling-condition.

EVENT: Disable enabling-condition-implies.

`;;; Ensures with Enabling`

DEFINITION:
e-ensures $(p,\ q,\ c,\ prg)$
$\leftrightarrow \quad \exists\ e\ ((e \in prg)$
$\qquad \wedge \quad$ enabling-condition $(c,\ e,\ prg)$
$\qquad \wedge \quad \forall\ old,\ new\ ((\mathrm{n}\,(old,\ new,\ e)$
$\qquad\qquad \wedge \quad \mathrm{eval}\,(\mathrm{list}\,(\texttt{'and},\ p,\ \mathrm{list}\,(\texttt{'not},\ q)),\ old))$
$\qquad\qquad \rightarrow \quad \mathrm{eval}\,(q,\ new)))$

EVENT: Disable e-ensures.

THEOREM: prove-e-ensures
$((e \in prg)$
$\wedge \quad$ enabling-condition $(c,\ e,\ prg)$
$\wedge \quad ((\mathrm{n}\,(\mathrm{oldee}\,(e,\ p,\ q),\ \mathrm{newee}\,(e,\ p,\ q),\ e)$
$\qquad \wedge \quad \mathrm{eval}\,(\mathrm{list}\,(\texttt{'and},\ p,\ \mathrm{list}\,(\texttt{'not},\ q)),\ \mathrm{oldee}\,(e,\ p,\ q)))$
$\qquad \rightarrow \quad \mathrm{eval}\,(q,\ \mathrm{newee}\,(e,\ p,\ q))))$
$\rightarrow \quad$ e-ensures $(p,\ q,\ c,\ prg)$

EVENT: Disable prove-e-ensures.

DEFINITION:
e-ensures-enabling $(p,\ q,\ c,\ prg)$

$$
\begin{aligned}
\leftrightarrow\quad \exists\, e\ ((e \in prg)\\
\wedge\quad \forall\, old,\ new\ ((\mathrm{n}\,(old,\ new,\ e) \rightarrow \mathrm{eval}\,(c,\ old))\\
\wedge\quad ((\mathrm{n}\,(old,\ new,\ e)\\
\wedge\quad \mathrm{eval}\,(\mathrm{list}\,(\texttt{'and},\ p,\ \mathrm{list}\,(\texttt{'not},\ q)),\ old))\\
\rightarrow\quad \mathrm{eval}\,(q,\ new)))\\
\wedge\quad \forall\, old\ (\mathrm{eval}\,(c,\ old) \rightarrow \exists\, new\ \mathrm{n}\,(old,\ new,\ e)))
\end{aligned}
$$

EVENT: Disable e-ensures-enabling.

THEOREM: help-prove-e-ensures

$$
\begin{aligned}
&((e \in prg)\\
&\ \wedge\quad (\mathrm{n}\,(\mathrm{oldeee}\,(c,\ e,\ p,\ q),\ \mathrm{neweee}\,(c,\ e,\ p,\ q),\ e)\\
&\qquad \rightarrow\quad (\mathrm{eval}\,(c,\ \mathrm{oldeee}\,(c,\ e,\ p,\ q)) \wedge \mathrm{eval}\,(q,\ \mathrm{neweee}\,(c,\ e,\ p,\ q))))\\
&\ \wedge\quad (\mathrm{eval}\,(c,\ \mathrm{oldeee\text{-}1}\,(c,\ e)) \rightarrow \mathrm{n}\,(\mathrm{oldeee\text{-}1}\,(c,\ e),\ new,\ e))\\
&\ \wedge\quad (\mathrm{eval}\,(c,\ \mathrm{oldeee}\,(c,\ e,\ p,\ q))\\
&\qquad \rightarrow\quad \mathrm{eval}\,(\mathrm{list}\,(\texttt{'and},\ p,\ \mathrm{list}\,(\texttt{'not},\ q)),\ \mathrm{oldeee}\,(c,\ e,\ p,\ q))))\\
&\rightarrow\quad \mathrm{e\text{-}ensures}\,(p,\ q,\ c,\ prg)
\end{aligned}
$$

THEOREM: e-ensures-implies

$$
\begin{aligned}
&(\mathrm{e\text{-}ensures}\,(p,\ q,\ c,\ prg) \rightarrow (\mathrm{eee}\,(c,\ p,\ prg,\ q) \in prg))\\
&\wedge\quad (\mathrm{e\text{-}ensures}\,(p,\ q,\ c,\ prg) \rightarrow \mathrm{enabling\text{-}condition}\,(c,\ \mathrm{eee}\,(c,\ p,\ prg,\ q),\ prg))\\
&\wedge\quad ((\mathrm{e\text{-}ensures}\,(p,\ q,\ c,\ prg)\\
&\qquad \wedge\quad \mathrm{n}\,(old,\ new,\ \mathrm{eee}\,(c,\ p,\ prg,\ q))\\
&\qquad \wedge\quad \mathrm{eval}\,(\mathrm{list}\,(\texttt{'and},\ p,\ \mathrm{list}\,(\texttt{'not},\ q)),\ old))\\
&\qquad \rightarrow\quad \mathrm{eval}\,(q,\ new))
\end{aligned}
$$

EVENT: Disable e-ensures-implies.

EVENT: Disable help-prove-e-ensures.

```
;;; Union Theorems
```

THEOREM: total-union-1
$\mathrm{total}\,(\mathrm{append}\,(prg\text{-}1,\ prg\text{-}2)) \rightarrow (\mathrm{total}\,(prg\text{-}1) \wedge \mathrm{total}\,(prg\text{-}2))$

THEOREM: total-union-2
$(\text{total}(\textit{prg-1}) \land \text{total}(\textit{prg-2})) \rightarrow \text{total}(\text{append}(\textit{prg-1}, \textit{prg-2}))$

THEOREM: total-union
$\text{total}(\text{append}(\textit{prg-1}, \textit{prg-2})) = (\text{total}(\textit{prg-1}) \land \text{total}(\textit{prg-2}))$

EVENT: Disable total-union-1.

EVENT: Disable total-union-2.

THEOREM: unless-union-1
$\text{unless}(p, q, \text{append}(\textit{prg-1}, \textit{prg-2}))$
$\rightarrow \quad (\text{unless}(p, q, \textit{prg-1}) \land \text{unless}(p, q, \textit{prg-2}))$

THEOREM: unless-union-2
$(\text{unless}(p, q, \textit{prg-1}) \land \text{unless}(p, q, \textit{prg-2}))$
$\rightarrow \quad \text{unless}(p, q, \text{append}(\textit{prg-1}, \textit{prg-2}))$

THEOREM: unless-union
$\text{unless}(p, q, \text{append}(\textit{prg-1}, \textit{prg-2}))$
$= \quad (\text{unless}(p, q, \textit{prg-1}) \land \text{unless}(p, q, \textit{prg-2}))$

EVENT: Disable unless-union-1.

EVENT: Disable unless-union-2.

THEOREM: ensures-union-1
$\text{ensures}(p, q, \text{append}(\textit{prg-1}, \textit{prg-2}))$
$\rightarrow \quad (\text{ensures}(p, q, \textit{prg-1}) \lor \text{ensures}(p, q, \textit{prg-2}))$

THEOREM: ensures-union-2
$(\text{ensures}(p, q, \textit{prg-1}) \lor \text{ensures}(p, q, \textit{prg-2}))$
$\rightarrow \quad \text{ensures}(p, q, \text{append}(\textit{prg-1}, \textit{prg-2}))$

EVENT: Disable ensures-union-1.

EVENT: Disable ensures-union-2.

THEOREM: ensures-union
$\text{ensures}(p, q, \text{append}(\textit{prg-1}, \textit{prg-2}))$
$= \quad (\text{ensures}(p, q, \textit{prg-1}) \lor \text{ensures}(p, q, \textit{prg-2}))$

```
;;; Help Prove Total, Unless, and Ensures.
```

DEFINITION:
total-sufficient $(statement, program, old, new)$
$= \quad ((statement \in program) \rightarrow \mathrm{n}(old, new, statement))$

THEOREM: help-prove-total
total-sufficient $(\mathrm{et}(prg), prg, \mathrm{oldt}(prg), new) \rightarrow \mathrm{total}(prg)$

DEFINITION:
unless-sufficient $(statement, program, old, new, p, q)$
$= \quad (((statement \in program)$
$\quad \wedge \quad \mathrm{n}(old, new, statement)$
$\quad \wedge \quad \mathrm{eval}(p, old)$
$\quad \wedge \quad (\neg\, \mathrm{eval}(q, old)))$
$\quad \rightarrow \quad \mathrm{eval}(\mathrm{list}(\texttt{'or}, p, q), new))$

THEOREM: help-prove-unless
unless-sufficient $(\mathrm{eu}(p, prg, q), prg, \mathrm{oldu}(p, prg, q), \mathrm{newu}(p, prg, q), p, q)$
$\rightarrow \quad \mathrm{unless}(p, q, prg)$

DEFINITION:
ensures-key $(statement, program, old, new, p, q)$
$= \quad ((statement \in program)$
$\quad \wedge \quad ((\mathrm{n}(old, new, statement) \wedge \mathrm{eval}(p, old) \wedge (\neg\, \mathrm{eval}(q, old)))$
$\quad\quad \rightarrow \quad \mathrm{eval}(q, new)))$

THEOREM: help-prove-ensures
ensures-key $(statement, prg, \mathrm{olde}(statement, p, q), \mathrm{newe}(statement, p, q), p, q)$
$\rightarrow \quad \mathrm{ensures}(p, q, prg)$

DEFINITION:
ensures-rest $(statement, key, program, old, new, p, q)$
$= \quad (((statement \in program)$
$\quad \wedge \quad (statement \neq key)$
$\quad \wedge \quad \mathrm{n}(old, new, statement)$
$\quad \wedge \quad \mathrm{eval}(p, old)$
$\quad \wedge \quad (\neg\, \mathrm{eval}(q, old)))$
$\quad \rightarrow \quad \mathrm{eval}(p, new))$

THEOREM: help-prove-unless-ensures
(ensures-key $(statement, prg, \mathrm{oldu}(p, prg, q), \mathrm{newu}(p, prg, q), p, q)$
$\wedge \quad$ ensures-rest $(\mathrm{eu}(p, prg, q),$

$statement,$
$prg,$
$\text{oldu}(p,\ prg,\ q),$
$\text{newu}(p,\ prg,\ q),$
$p,$
$q))$
$\rightarrow\quad \text{unless}(p,\ q,\ prg)$

`;;; Strengthening and Weakening Unless and Ensures`

DEFINITION:
$\text{stronger-p}(p,\ q) \leftrightarrow \forall\, state\ (\text{eval}(p,\ state) \rightarrow \text{eval}(q,\ state))$

EVENT: Disable stronger-p.

THEOREM: stronger-p-implies
$(\text{stronger-p}(p,\ q) \wedge \text{eval}(p,\ state)) \rightarrow \text{eval}(q,\ state)$

THEOREM: stronger-p-rewrite
$\text{stronger-p}(p,\ q) = (\text{eval}(p,\ \text{states}(p,\ q)) \rightarrow \text{eval}(q,\ \text{states}(p,\ q)))$

EVENT: Disable stronger-p-implies.

DEFINITION:
$\text{equal-p}(p,\ q) \leftrightarrow \forall\, state\ (\text{eval}(p,\ state) = \text{eval}(q,\ state))$

EVENT: Disable equal-p.

THEOREM: equal-p-implies
$\text{equal-p}(p,\ q) \rightarrow (\text{eval}(p,\ state) = \text{eval}(q,\ state))$

EVENT: Disable equal-p-implies.

THEOREM: equal-p-rewrite
$\text{equal-p}(p,\ q) = (\text{eval}(p,\ \text{statee}(p,\ q)) = \text{eval}(q,\ \text{statee}(p,\ q)))$

THEOREM: equal-p-commutative
$\text{equal-p}(p,\ q) = \text{equal-p}(q,\ p)$

THEOREM: ensures-strengthen-left
$(\text{ensures}(q,\ r,\ prg) \wedge \text{stronger-p}(p,\ q)) \rightarrow \text{ensures}(p,\ r,\ prg)$

EVENT: Disable ensures-strengthen-left.

THEOREM: ensures-weaken-right
(ensures $(p,\ q,\ prg)$ $\wedge$ stronger-p $(q,\ r)$) $\rightarrow$ ensures $(p,\ r,\ prg)$

EVENT: Disable ensures-weaken-right.

THEOREM: unless-weaken-right
(unless $(p,\ q,\ prg)$ $\wedge$ stronger-p $(q,\ r)$) $\rightarrow$ unless $(p,\ r,\ prg)$

EVENT: Disable unless-weaken-right.

THEOREM: unless-equal-p
equal-p $(p,\ q)$ $\rightarrow$ (unless $(p,\ r,\ prg)$ = unless $(q,\ r,\ prg)$)

THEOREM: unless-conjunction
(unless $(p\text{-}1,\ q,\ prg)$ $\wedge$ unless $(p\text{-}2,\ q,\ prg)$ $\wedge$ equal-p (p, list (`'and`, $p\text{-}1$, $p\text{-}2$)))
$\rightarrow$ unless $(p,\ q,\ prg)$

THEOREM: unless-disjunction
(unless $(p\text{-}1,\ q,\ prg)$ $\wedge$ unless $(p\text{-}2,\ q,\ prg)$ $\wedge$ equal-p (p, list (`'or`, $p\text{-}1$, $p\text{-}2$)))
$\rightarrow$ unless $(p,\ q,\ prg)$

```
;;; Fairness Theorems
```

THEOREM: unconditional-fairness
(unless $(p,\ q,\ prg)$ $\wedge$ ensures $(p,\ q,\ prg)$ $\wedge$ total (prg)) $\rightarrow$ leads-to $(p,\ q,\ prg)$

EVENT: Disable unconditional-fairness.

THEOREM: unconditional-fairness-general
(unless $(p\text{-}1,\ q\text{-}1,\ prg)$
$\wedge$ ensures $(p\text{-}2,\ q\text{-}2,\ prg)$
$\wedge$ (eval (p, s (prg, ileads $(p,\ prg,\ q)$)) $\rightarrow$ eval ($p\text{-}1$, s (prg, ileads $(p,\ prg,\ q)$)))
$\wedge$ stronger-p $(p\text{-}1,\ p\text{-}2)$
$\wedge$ (eval ($q\text{-}1$, s (prg, jleads (ileads $(p\text{-}1,\ prg,\ q)$, prg, $q\text{-}1$)))
$\rightarrow$ eval (q, s (prg, jleads (ileads $(p\text{-}1,\ prg,\ q)$, prg, $q\text{-}1$))))
$\wedge$ stronger-p $(q\text{-}2,\ q\text{-}1)$
$\wedge$ total (prg))
$\rightarrow$ leads-to $(p,\ q,\ prg)$

EVENT: Disable unconditional-fairness-general.

CONSERVATIVE AXIOM: strong-fairness

$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{unless}\,(p,\ q,\ prg)} \\
\ \wedge & \text{e-ensures}\,(p,\ q,\ c,\ prg) \\
\ \wedge & \text{leads-to}\,(p,\ \text{list}\,(\texttt{'or},\ q,\ c),\ prg) \\
\ \wedge & \text{strongly-fair}\,(prg)) \\
\rightarrow & \text{leads-to}\,(p,\ q,\ prg)
\end{array}
$$

Simultaneously, we introduce the new function symbol *strongly-fair*.

EVENT: Disable strong-fairness.

THEOREM: strong-fairness-general

$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{unless}\,(p\text{-}1,\ q\text{-}1,\ prg)} \\
\ \wedge & \text{e-ensures}\,(p\text{-}2,\ q\text{-}2,\ c,\ prg) \\
\ \wedge & \text{leads-to}\,(p,\ \text{list}\,(\texttt{'or},\ q,\ c),\ prg) \\
\ \wedge & (\text{eval}\,(p,\ \text{s}\,(prg,\ \text{ileads}\,(p,\ prg,\ q))) \rightarrow \text{eval}\,(p\text{-}1,\ \text{s}\,(prg,\ \text{ileads}\,(p,\ prg,\ q)))) \\
\ \wedge & (\text{eval}\,(p\text{-}1,\ \text{s}\,(prg,\ \text{ileads}\,(p\text{-}1,\ prg,\ \text{list}\,(\texttt{'or},\ q\text{-}1,\ c)))) \\
 & \quad \rightarrow \quad \text{eval}\,(p,\ \text{s}\,(prg,\ \text{ileads}\,(p\text{-}1,\ prg,\ \text{list}\,(\texttt{'or},\ q\text{-}1,\ c))))) \\
\ \wedge & \text{stronger-p}\,(p\text{-}1,\ p\text{-}2) \\
\ \wedge & (\text{eval}\,(q\text{-}1,\ \text{s}\,(prg,\ \text{jleads}\,(\text{ileads}\,(p\text{-}1,\ prg,\ q),\ prg,\ q\text{-}1))) \\
 & \quad \rightarrow \quad \text{eval}\,(q,\ \text{s}\,(prg,\ \text{jleads}\,(\text{ileads}\,(p\text{-}1,\ prg,\ q),\ prg,\ q\text{-}1)))) \\
\ \wedge & (\text{eval}\,(\text{list}\,(\texttt{'or},\ q,\ c), \\
 & \qquad\quad \text{s}\,(prg, \\
 & \qquad\qquad \text{jleads}\,(\text{ileads}\,(p,\ prg,\ \text{list}\,(\texttt{'or},\ q\text{-}1,\ c)),\ prg,\ \text{list}\,(\texttt{'or},\ q,\ c)))) \\
 & \quad \rightarrow \quad \text{eval}\,(\text{list}\,(\texttt{'or},\ q\text{-}1,\ c), \\
 & \qquad\qquad\quad \text{s}\,(prg, \\
 & \qquad\qquad\qquad \text{jleads}\,(\text{ileads}\,(p,\ prg,\ \text{list}\,(\texttt{'or},\ q\text{-}1,\ c)), \\
 & \qquad\qquad\qquad\qquad\qquad prg, \\
 & \qquad\qquad\qquad\qquad\qquad \text{list}\,(\texttt{'or},\ q,\ c))))) \\
\ \wedge & \text{stronger-p}\,(q\text{-}2,\ q\text{-}1) \\
\ \wedge & \text{strongly-fair}\,(prg)) \\
\rightarrow & \text{leads-to}\,(p,\ q,\ prg)
\end{array}
$$

EVENT: Disable strong-fairness-general.

DEFINITION:

$$
\begin{array}{ll}
\multicolumn{2}{l}{\text{wfw}\,(i,\ j,\ p,\ q,\ c,\ prg)} \\
= & \textbf{if}\ i < j \\
 & \textbf{then if}\ \text{eval}\,(q,\ \text{s}\,(prg,\ i))\ \ \textbf{then}\ i \\
 & \qquad \textbf{elseif}\ \text{eval}\,(p,\ \text{s}\,(prg,\ i)) \\
 & \qquad \textbf{then if}\ \text{eval}\,(c,\ \text{s}\,(prg,\ i))\ \ \textbf{then}\ \text{wfw}\,(1 + i,\ j,\ p,\ q,\ c,\ prg) \\
 & \qquad\qquad \textbf{else}\ i\ \textbf{endif} \\
 & \qquad \textbf{else}\ i\ \textbf{endif} \\
 & \textbf{else}\ \text{fix}\,(i)\ \textbf{endif}
\end{array}
$$

THEOREM: wfw-bigger
$\text{wfw}(i, j, p, q, c, prg) \not< i$

THEOREM: about-wfw
$(\text{unless}(p, q, prg) \land \text{listp}(prg) \land \text{eval}(p, \text{s}(prg, i)) \land (j \not< i))$
$\rightarrow \quad (\text{eval}(q, \text{s}(prg, \text{wfw}(i, j, p, q, c, prg)))$
$\quad \lor \quad (\text{eval}(p, \text{s}(prg, \text{wfw}(i, j, p, q, c, prg)))$
$\quad\quad \land \quad (\neg\, \text{eval}(q, \text{s}(prg, \text{wfw}(i, j, p, q, c, prg))))$
$\quad\quad \land \quad (\neg\, \text{eval}(c, \text{s}(prg, \text{wfw}(i, j, p, q, c, prg)))))$
$\quad \lor \quad ((\text{wfw}(i, j, p, q, c, prg) = \text{fix}(j))$
$\quad\quad \land \quad (\text{eval}(p, \text{s}(prg, j)) \lor \text{eval}(q, \text{s}(prg, j)))))$

DEFINITION:
$\text{witness}(p, q, c, prg)$
$= \quad \text{wfw}(\text{ileads}(p, prg, q),$
$\quad\quad \text{next}(prg, \text{eee}(c, p, prg, q), \text{ileads}(p, prg, q)),$
$\quad\quad p,$
$\quad\quad q,$
$\quad\quad c,$
$\quad\quad prg)$

THEOREM: weak-fairness
$(\text{unless}(p, q, prg)$
$\land \quad \text{e-ensures}(p, q, c, prg)$
$\land \quad (\text{eval}(\text{list}(\texttt{'and}, p, \text{list}(\texttt{'not}, q)), \text{s}(prg, \text{witness}(p, q, c, prg)))$
$\quad \rightarrow \quad \text{eval}(c, \text{s}(prg, \text{witness}(p, q, c, prg)))))$
$\rightarrow \quad \text{leads-to}(p, q, prg)$

EVENT: Disable weak-fairness.

THEOREM: weak-fairness-general
$(\text{unless}(p\text{-}1, q\text{-}1, prg)$
$\land \quad \text{e-ensures}(p\text{-}2, q\text{-}2, c, prg)$
$\land \quad (\text{eval}(\text{list}(\texttt{'and}, p\text{-}1, \text{list}(\texttt{'not}, q\text{-}1)), \text{s}(prg, \text{witness}(p\text{-}1, q\text{-}1, c, prg)))$
$\quad \rightarrow \quad \text{eval}(c, \text{s}(prg, \text{witness}(p\text{-}1, q\text{-}1, c, prg))))$
$\land \quad (\text{eval}(p, \text{s}(prg, \text{ileads}(p, prg, q))) \rightarrow \text{eval}(p\text{-}1, \text{s}(prg, \text{ileads}(p, prg, q))))$
$\land \quad \text{stronger-p}(p\text{-}1, p\text{-}2)$
$\land \quad (\text{eval}(q\text{-}1, \text{s}(prg, \text{jleads}(\text{ileads}(p\text{-}1, prg, q), prg, q\text{-}1)))$
$\quad \rightarrow \quad \text{eval}(q, \text{s}(prg, \text{jleads}(\text{ileads}(p\text{-}1, prg, q), prg, q\text{-}1))))$
$\land \quad \text{stronger-p}(q\text{-}2, q\text{-}1))$
$\rightarrow \quad \text{leads-to}(p, q, prg)$

EVENT: Disable weak-fairness-general.

EVENT: Disable witness.

THEOREM: deadlock-freedom-witness
$$\begin{array}{l}
(\text{unless}\,(inv,\ \texttt{'(false)},\ prg) \\
\ \wedge\ \ \text{enabling-condition}\,(c,\ e,\ prg) \\
\ \wedge\ \ (\text{eval}\,(inv,\ \text{s}\,(prg,\ \text{next}\,(prg,\ e,\ \text{ileads}\,(inv,\ prg,\ \texttt{'(false)})))) \\
\qquad \rightarrow\ \ (\neg\ \text{eval}\,(c,\ \text{s}\,(prg,\ \text{next}\,(prg,\ e,\ \text{ileads}\,(inv,\ prg,\ \texttt{'(false)})))))) \\
\ \wedge\ \ \text{schedulable}\,(prg)) \\
\rightarrow\ \ \text{leads-to}\,(inv,\ \texttt{'(false)},\ prg)
\end{array}$$

EVENT: Disable deadlock-freedom-witness.

CONSERVATIVE AXIOM: deadlock-freedom
$$\begin{array}{l}
(\text{unless}\,(inv,\ \texttt{'(false)},\ prg) \\
\ \wedge\ \ \text{enabling-condition}\,(c,\ e,\ prg) \\
\ \wedge\ \ (\text{eval}\,(inv,\ \text{s}\,(prg,\ \text{next}\,(prg,\ e,\ \text{ileads}\,(inv,\ prg,\ \texttt{'(false)})))) \\
\qquad \rightarrow\ \ (\neg\ \text{eval}\,(c,\ \text{s}\,(prg,\ \text{next}\,(prg,\ e,\ \text{ileads}\,(inv,\ prg,\ \texttt{'(false)})))))) \\
\ \wedge\ \ \text{deadlock-free}\,(prg)) \\
\rightarrow\ \ \text{leads-to}\,(inv,\ \texttt{'(false)},\ prg)
\end{array}$$

Simultaneously, we introduce the new function symbol *deadlock-free*.

THEOREM: deadlock-freedom-general
$$\begin{array}{l}
(\text{unless}\,(inv,\ \texttt{'(false)},\ prg) \\
\ \wedge\ \ \text{enabling-condition}\,(c,\ e,\ prg) \\
\ \wedge\ \ (\text{eval}\,(inv,\ \text{s}\,(prg,\ \text{next}\,(prg,\ e,\ \text{ileads}\,(inv,\ prg,\ \texttt{'(false)})))) \\
\qquad \rightarrow\ \ (\neg\ \text{eval}\,(c,\ \text{s}\,(prg,\ \text{next}\,(prg,\ e,\ \text{ileads}\,(inv,\ prg,\ \texttt{'(false)})))))) \\
\ \wedge\ \ ((\neg\ \text{eval}\,(inv,\ \text{s}\,(prg,\ \text{ii}\,(p,\ prg)))) \rightarrow \text{eval}\,(p,\ \text{s}\,(prg,\ \text{ii}\,(p,\ prg)))) \\
\ \wedge\ \ \text{deadlock-free}\,(prg)) \\
\rightarrow\ \ \text{invariant}\,(p,\ prg)
\end{array}$$

EVENT: Enable eval.

EVENT: Enable n.

```
;;; Some Helpful Definitions and Theorems
```

DEFINITION:
$$\begin{array}{l}
\text{update-assoc}\,(key,\ value,\ alist) \\
=\ \ \textbf{if}\ \text{listp}\,(alist) \\
\qquad \textbf{then if}\ \text{caar}\,(alist) = key\ \ \textbf{then}\ \text{cons}\,(\text{cons}\,(key,\ value),\ \text{cdr}\,(alist)) \\
\qquad\qquad \textbf{else}\ \text{cons}\,(\text{car}\,(alist),\ \text{update-assoc}\,(key,\ value,\ \text{cdr}\,(alist)))\ \textbf{endif} \\
\qquad \textbf{else}\ \text{list}\,(\text{cons}\,(key,\ value))\ \textbf{endif}
\end{array}$$

THEOREM: simplify-assoc
assoc (*key-1*, update-assoc (*key-2*, *value*, *alist*))
$=$ **if** *key-1* $=$ *key-2* **then** cons (*key-1*, *value*)
else assoc (*key-1*, *alist*) **endif**

DEFINITION:
add1-mod (n, x)
$=$ **if** $(1 + x) < n$ **then** $1 + x$
else 0 **endif**

DEFINITION:
sub1-mod (n, x)
$=$ **if** $x < n$
then if $x \simeq 0$ **then** $n - 1$
else $x - 1$ **endif**
else 0 **endif**

DEFINITION:
uc (*old*, *new*, *keys*, *excpt*)
$=$ **if** listp (*keys*)
then if car (*keys*) $\in$ *excpt* **then** uc (*old*, *new*, cdr (*keys*), *excpt*)
elseif assoc (car (*keys*), *old*) $=$ assoc (car (*keys*), *new*)
then uc (*old*, *new*, cdr (*keys*), *excpt*)
else f endif
else t endif

THEOREM: uc-basic-property
(uc (*old*, *new*, *keys*, *excpt*) $\land$ (*key* $\in$ *keys*) $\land$ (*key* $\notin$ *excpt*))
$\rightarrow$ ((assoc (*key*, *old*) $=$ assoc (*key*, *new*)) $=$ **t**)

THEOREM: uc-commutative
uc (*old*, *new*, *keys*, *excpt*) $=$ uc (*new*, *old*, *keys*, *excpt*)

THEOREM: uc-reflexive
uc (*list*, *list*, *keys*, *excpt*)

THEOREM: uc-of-update-assoc
uc (*list-1*, update-assoc (*key*, *value*, *list-2*), *keys*, *excpt*)
$=$ **if** *key* $\in$ *excpt* **then** uc (*list-1*, *list-2*, *keys*, *excpt*)
elseif *key* $\in$ *keys*
then (assoc (*key*, *list-1*) $=$ cons (*key*, *value*))
$\land$ uc (*list-1*, *list-2*, *keys*, cons (*key*, *excpt*))
else uc (*list-1*, *list-2*, *keys*, *excpt*) **endif**

THEOREM: strip-cars-append
strip-cars (append (a, b)) $=$ append (strip-cars (a), strip-cars (b))

THEOREM: uc-append
$\text{uc}\,(old,\ new,\ \text{append}\,(a,\ b),\ excpt)$
$= \quad (\text{uc}\,(old,\ new,\ a,\ excpt) \land \text{uc}\,(old,\ new,\ b,\ excpt))$

THEOREM: uc-commutative-2
$\text{uc}\,(old,\ new,\ \text{append}\,(a,\ b),\ excpt) = \text{uc}\,(old,\ new,\ \text{append}\,(b,\ a),\ excpt)$

EVENT: Disable uc-append.

THEOREM: key-not-member-strip-cars
$(key \not\in \text{strip-cars}\,(alist)) \rightarrow (\text{assoc}\,(key,\ alist) = \mathbf{f})$

THEOREM: uc-property
$(\text{uc}\,(old,\ new,\ \text{append}\,(\text{strip-cars}\,(old),\ \text{strip-cars}\,(new)),\ excpt)$
$\land \quad (key \not\in excpt))$
$\rightarrow \quad ((\text{assoc}\,(key,\ old) = \text{assoc}\,(key,\ new)) = \mathbf{t})$

THEOREM: about-uc
$(\text{uc}\,(a,\ b,\ \text{append}\,(\text{strip-cars}\,(a),\ \text{strip-cars}\,(b)),\ excpt) \land (key \not\in excpt))$
$\rightarrow \quad (\text{assoc}\,(key,\ a) = \text{assoc}\,(key,\ b))$

DEFINITION:
$\text{changed}\,(old,\ new,\ excpt) = \text{uc}\,(old,\ new,\ \text{strip-cars}\,(\text{append}\,(old,\ new)),\ excpt)$

EVENT: Make the library `"interpreter"`.

Index