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

`;;; The Program`

DEFINITION:
status (*state*, *index*) = cdr (assoc (cons (`'s`, *index*), *state*))

DEFINITION:
thinking (*state*, *index*) = (status (*state*, *index*) = `'thinking`)

DEFINITION:
hungry (*state*, *index*) = (status (*state*, *index*) = `'hungry`)

DEFINITION:
eating (*state*, *index*) = (status (*state*, *index*) = `'eating`)

DEFINITION:
fork (*state*, *index*) = cdr (assoc (cons (`'f`, *index*), *state*))

DEFINITION: free (*state*, *index*) = (fork (*state*, *index*) = `'free`)

DEFINITION:
owns-left (*state*, *index*) = (fork (*state*, *index*) = *index*)

DEFINITION:
owns-right (*state*, *index*, *n*) = (fork (*state*, add1-mod (*n*, *index*)) = *index*)

DEFINITION:
thinking-to (*old*, *new*, *index*)
= **if** thinking (*old*, *index*)
  then (thinking (*new*, *index*) ∨ hungry (*new*, *index*))
    ∧ changed (*old*, *new*, list (cons (`'s`, *index*)))
  else changed (*old*, *new*, **nil**) **endif**

DEFINITION:
hungry-left (*old*, *new*, *index*)
= (hungry (*old*, *index*)
  ∧ free (*old*, *index*)
  ∧ owns-left (*new*, *index*)
  ∧ changed (*old*, *new*, list (cons (`'f`, *index*))))

DEFINITION:
hungry-right (*old*, *new*, *index*, *n*)
= (hungry (*old*, *index*)
  ∧ free (*old*, add1-mod (*n*, *index*))
  ∧ owns-right (*new*, *index*, *n*)
  ∧ changed (*old*, *new*, list (cons (`'f`, add1-mod (*n*, *index*)))))

DEFINITION:
hungry-both (*old*, *new*, *index*, *n*)
= **if** hungry (*old*, *index*)
 ∧ owns-left (*old*, *index*)
 ∧ owns-right (*old*, *index*, *n*)
then eating (*new*, *index*) ∧ changed (*old*, *new*, list (cons (`'s`, *index*)))
else changed (*old*, *new*, **nil**) **endif**

DEFINITION:
eating-to (*old*, *new*, *index*, *n*)
= **if** eating (*old*, *index*)
then thinking (*new*, *index*)
 ∧ free (*new*, *index*)
 ∧ free (*new*, add1-mod (*n*, *index*))
 ∧ changed (*old*,
 new,
 list (cons (`'s`, *index*),
 cons (`'f`, *index*),
 cons (`'f`, add1-mod (*n*, *index*))))
else changed (*old*, *new*, **nil**) **endif**

DEFINITION:
phil (*index*, *n*)
= list (list (`'thinking-to`, *index*),
 list (`'hungry-left`, *index*),
 list (`'hungry-right`, *index*, *n*),
 list (`'hungry-both`, *index*, *n*),
 list (`'eating-to`, *index*, *n*))

DEFINITION:
ring (*index*, *n*)
= **if** *index* ≃ `0` **then nil**
else append (phil (*index* − 1, *n*), ring (*index* − 1, *n*)) **endif**

DEFINITION: phil-prg (*n*) = ring (*n*, *n*)

EVENT: Disable phil-prg.

EVENT: Disable *1*phil-prg.

`;;; Correctness`

THEOREM: member-ring

$$
\begin{aligned}
&(\mathit{statement} \in \mathrm{ring}\,(\mathit{index},\ n))\\
=\ \ &((\mathit{statement} \in \mathrm{phil}\,(\mathrm{cadr}\,(\mathit{statement}),\ n))\\
&\quad \wedge\ \ (\mathrm{cadr}\,(\mathit{statement}) \in \mathbf{N})\\
&\quad \wedge\ \ (\mathrm{cadr}\,(\mathit{statement}) < \mathit{index}))
\end{aligned}
$$

THEOREM: member-phil-prg

$$
\begin{aligned}
&(\mathit{statement} \in \text{phil-prg}\,(n))\\
=\ \ &((\mathit{statement} \in \mathrm{phil}\,(\mathrm{cadr}\,(\mathit{statement}),\ n))\\
&\quad \wedge\ \ (\mathrm{cadr}\,(\mathit{statement}) \in \mathbf{N})\\
&\quad \wedge\ \ (\mathrm{cadr}\,(\mathit{statement}) < n))
\end{aligned}
$$

DEFINITION:

$$
\begin{aligned}
&\text{proper-forks-rec}\,(\mathit{state},\ \mathit{index},\ n)\\
=\ \ &\textbf{if } \mathit{index} \simeq \mathtt{0} \textbf{ then t}\\
&\textbf{else } (\mathrm{free}\,(\mathit{state},\ \mathit{index} - 1)\\
&\qquad\quad \vee\ \ (\mathrm{fork}\,(\mathit{state},\ \mathit{index} - 1) = (\mathit{index} - 1))\\
&\qquad\quad \vee\ \ (\mathrm{fork}\,(\mathit{state},\ \mathit{index} - 1) = \text{sub1-mod}\,(n,\ \mathit{index} - 1)))\\
&\qquad \wedge\ \ \text{proper-forks-rec}\,(\mathit{state},\ \mathit{index} - 1,\ n)\ \textbf{endif}
\end{aligned}
$$

THEOREM: proper-forks-rec-implies

$$
\begin{aligned}
&(\text{proper-forks-rec}\,(\mathit{state},\ \mathit{index},\ n)\\
&\ \wedge\ \ (n \not< \mathit{index})\\
&\ \wedge\ \ (\mathtt{1} < n)\\
&\ \wedge\ \ (i < \mathit{index})\\
&\ \wedge\ \ (i \in \mathbf{N}))\\
\rightarrow\ \ &(\mathrm{free}\,(\mathit{state},\ i)\\
&\quad \vee\ \ (\mathrm{fork}\,(\mathit{state},\ i) = i)\\
&\quad \vee\ \ (\mathrm{fork}\,(\mathit{state},\ i) = \text{sub1-mod}\,(n,\ i)))
\end{aligned}
$$

DEFINITION: $\text{proper-forks}\,(\mathit{state},\ n) = \text{proper-forks-rec}\,(\mathit{state},\ n,\ n)$

THEOREM: proper-forks-implies

$$
\begin{aligned}
&(\text{proper-forks}\,(\mathit{state},\ n) \wedge (\mathtt{1} < n) \wedge (i < n) \wedge (i \in \mathbf{N}))\\
\rightarrow\ \ &(\mathrm{free}\,(\mathit{state},\ i)\\
&\quad \vee\ \ (\mathrm{fork}\,(\mathit{state},\ i) = i)\\
&\quad \vee\ \ (\mathrm{fork}\,(\mathit{state},\ i) = \text{sub1-mod}\,(n,\ i)))
\end{aligned}
$$

DEFINITION:

$$
\begin{aligned}
&\text{proper-phil}\,(\mathit{state},\ \mathit{phil},\ \mathit{right})\\
=\ \ &((\mathrm{thinking}\,(\mathit{state},\ \mathit{phil})\\
&\quad \rightarrow\ \ ((\mathrm{fork}\,(\mathit{state},\ \mathit{phil}) \neq \mathit{phil})\\
&\qquad\quad \wedge\ \ (\mathrm{fork}\,(\mathit{state},\ \mathit{right}) \neq \mathit{phil})))\\
&\ \wedge\ \ (\mathrm{eating}\,(\mathit{state},\ \mathit{phil})\\
&\qquad \rightarrow\ \ ((\mathrm{fork}\,(\mathit{state},\ \mathit{phil}) = \mathit{phil})
\end{aligned}
$$

$$
\begin{array}{l}
\qquad\qquad\quad \wedge \quad (\text{fork}\,(state,\ right) = phil)))\\
\quad \wedge \quad (\text{thinking}\,(state,\ phil)\\
\qquad\quad \vee \quad \text{hungry}\,(state,\ phil)\\
\qquad\quad \vee \quad \text{eating}\,(state,\ phil)))
\end{array}
$$

Definition:

$$
\begin{array}{l}
\text{proper-phils-rec}\,(state,\ index,\ n)\\
= \quad \textbf{if } index \simeq \mathtt{0} \textbf{ then t}\\
\qquad \textbf{else } \text{proper-phil}\,(state,\ index - 1,\ \text{add1-mod}\,(n,\ index - 1))\\
\qquad\qquad \wedge \quad \text{proper-phils-rec}\,(state,\ index - 1,\ n) \textbf{ endif}
\end{array}
$$

Theorem: proper-phils-rec-implies-proper-phil

$$
\begin{array}{l}
(\text{proper-phils-rec}\,(state,\ index,\ n)\\
\ \wedge \quad (\mathtt{1} < n)\\
\ \wedge \quad (n \not< index)\\
\ \wedge \quad (phil < index)\\
\ \wedge \quad (phil \in \mathbf{N})\\
\ \wedge \quad (right = \text{add1-mod}\,(n,\ phil)))\\
\rightarrow \quad \text{proper-phil}\,(state,\ phil,\ right)
\end{array}
$$

Theorem: proper-phils-rec-implies

$$
\begin{array}{l}
(\text{proper-phils-rec}\,(state,\ index,\ n)\\
\ \wedge \quad (n \not< index)\\
\ \wedge \quad (\mathtt{1} < n)\\
\ \wedge \quad (phil < index)\\
\ \wedge \quad (phil \in \mathbf{N}))\\
\rightarrow \quad ((\text{thinking}\,(state,\ phil)\\
\qquad\quad \rightarrow \quad ((\neg\ \text{owns-left}\,(state,\ phil))\\
\qquad\qquad\quad \wedge \quad (\neg\ \text{owns-right}\,(state,\ phil,\ n))))\\
\qquad \wedge \quad (\text{eating}\,(state,\ phil)\\
\qquad\qquad \rightarrow \quad (\text{owns-left}\,(state,\ phil) \wedge \text{owns-right}\,(state,\ phil,\ n)))\\
\qquad \wedge \quad (\text{thinking}\,(state,\ phil)\\
\qquad\qquad \vee \quad \text{hungry}\,(state,\ phil)\\
\qquad\qquad \vee \quad \text{eating}\,(state,\ phil)))
\end{array}
$$

Definition: $\text{proper-phils}\,(state,\ n) = \text{proper-phils-rec}\,(state,\ n,\ n)$

Theorem: proper-phils-implies-proper-phil

$$
\begin{array}{l}
(\text{proper-phils}\,(state,\ n)\\
\ \wedge \quad (\mathtt{1} < n)\\
\ \wedge \quad (phil < n)\\
\ \wedge \quad (phil \in \mathbf{N})\\
\ \wedge \quad (right = \text{add1-mod}\,(n,\ phil)))\\
\rightarrow \quad \text{proper-phil}\,(state,\ phil,\ right)
\end{array}
$$

DEFINITION:
all-lefts (*state*, *index*)
$=$ **if** *index* $\simeq$ `0` **then t**
else owns-left (*state*, *index* $-$ 1)
$\land$ hungry (*state*, *index* $-$ 1)
$\land$ all-lefts (*state*, *index* $-$ 1) **endif**

THEOREM: all-lefts-implies
$((i < n) \land (i \in \mathbf{N}) \land$ all-lefts (*state*, n))
$\rightarrow$ (hungry (*state*, i) $\land$ owns-left (*state*, i))

DEFINITION:
all-rights-rec (*state*, *index*, n)
$=$ **if** *index* $\simeq$ `0` **then t**
else owns-right (*state*, *index* $-$ 1, n)
$\land$ hungry (*state*, *index* $-$ 1)
$\land$ all-rights-rec (*state*, *index* $-$ 1, n) **endif**

THEOREM: all-rights-rec-implies
$((1 < n)$
$\land$ $(n \not< \mathit{index})$
$\land$ $(i < \mathit{index})$
$\land$ $(i \in \mathbf{N})$
$\land$ all-rights-rec (*state*, *index*, n))
$\rightarrow$ (hungry (*state*, i) $\land$ owns-right (*state*, i, n))

DEFINITION: all-rights (*state*, n) = all-rights-rec (*state*, n, n)

THEOREM: all-rights-implies
$((1 < n) \land (i < n) \land (i \in \mathbf{N}) \land$ all-rights (*state*, n))
$\rightarrow$ (hungry (*state*, i) $\land$ owns-right (*state*, i, n))

```
;(defn initial (state n)
;  (and (proper-forks state n)
;       (proper-phils state n)))
```

THEOREM: listp-phil-prg
listp (phil-prg (n)) = $(n \not\simeq 0)$

DEFINITION:
pfusi (*index*, n, *statement*)
$=$ **if** *index* $\simeq$ `0` **then t**
elseif (*index* $-$ 1) = cadr (*statement*)
then pfusi (*index* $-$ 1, n, *statement*)
elseif (*index* $-$ 1) = add1-mod (n, cadr (*statement*))
then pfusi (*index* $-$ 1, n, *statement*)
else pfusi (*index* $-$ 1, n, *statement*) **endif**

THEOREM: proper-forks-rec-unless-sufficient

$((n \not< \mathit{index}) \land (\mathtt{1} < n))$
$\rightarrow$ unless-sufficient (*statement*,
phil-prg (n),
old,
new,
`` `(proper-forks-rec state ',index ',n) ``,
`'(false)`)

DEFINITION:

proper-triple (*state*, *phil*, n)
$=$ (proper-phil (*state*, sub1-mod (n, *phil*), add1-mod (n, sub1-mod (n, *phil*)))
$\land$ proper-phil (*state*, *phil*, add1-mod (n, *phil*))
$\land$ proper-phil (*state*,
add1-mod (n, *phil*),
add1-mod (n, add1-mod (n, *phil*))))

DEFINITION:

but-triple (*state*, *index*, *phil*, n)
$=$ **if** *index* $\simeq \mathtt{0}$ **then t**
elseif $((\mathit{index} - 1) = $ sub1-mod (n, *phil*))
$\lor$ $((\mathit{index} - 1) = \mathit{phil})$
$\lor$ $((\mathit{index} - 1) = $ add1-mod (n, *phil*))
then but-triple (*state*, $\mathit{index} - 1$, *phil*, n)
else proper-phil (*state*, $\mathit{index} - 1$, add1-mod (n, $\mathit{index} - 1$))
$\land$ but-triple (*state*, $\mathit{index} - 1$, *phil*, n) **endif**

THEOREM: lessp-1

$(\mathtt{1} < n) = ((n \not\simeq \mathtt{0}) \land (n \neq \mathtt{1}))$

THEOREM: lessp-2

$(x < \mathtt{2}) = ((x = \mathtt{1}) \lor (x \simeq \mathtt{0}))$

THEOREM: lessp-2-2

$(\mathtt{2} < x) = ((x \not\simeq \mathtt{0}) \land (x \neq \mathtt{1}) \land (x \neq \mathtt{2}))$

THEOREM: proper-triple-preserved

$((\mathtt{1} < n)$
$\land$ $(\mathit{phil} < n)$
$\land$ $(\mathit{phil} \in \mathbf{N})$
$\land$ (*statement* $\in$ phil (*phil*, n))
$\land$ n (*old*, *new*, *statement*)
$\land$ proper-triple (*old*, *phil*, n))
$\rightarrow$ proper-triple (*new*, *phil*, n)

THEOREM: but-triple-preserved

```
((1 < n)
 ∧  (n ≮ index)
 ∧  (phil < n)
 ∧  (phil ∈ N)
 ∧  (statement ∈ phil (phil, n))
 ∧  n (old, new, statement)
 ∧  but-triple (old, index, phil, n))
→   but-triple (new, index, phil, n)
```

THEOREM: but-triple-and-triple-all

```
((phil ∈ N) ∧ (n ≮ index))
→   (proper-phils-rec (state, index, n)
     =   (but-triple (state, index, phil, n)
          ∧   if add1-mod (n, phil) < index
              then if phil < index
                   then if sub1-mod (n, phil) < index
                        then proper-triple (state, phil, n)
                        else proper-phil (state,
                                          phil,
                                          add1-mod (n, phil))
                             ∧   proper-phil (state,
                                              add1-mod (n,
                                                        phil),
                                              add1-mod (n,
                                                        add1-mod (n,
                                                                  phil))) endif
                   elseif sub1-mod (n, phil) < index
                   then proper-phil (state,
                                     sub1-mod (n, phil),
                                     add1-mod (n, sub1-mod (n, phil)))
                        ∧   proper-phil (state,
                                         add1-mod (n, phil),
                                         add1-mod (n,
                                                   add1-mod (n, phil)))
                   else proper-phil (state,
                                     add1-mod (n, phil),
                                     add1-mod (n,
                                               add1-mod (n, phil))) endif
              elseif phil < index
              then if sub1-mod (n, phil) < index
                   then proper-phil (state, phil, add1-mod (n, phil))
                        ∧   proper-phil (state,
                                         sub1-mod (n, phil),
```

```
                                        add1-mod (n,
                                                  sub1-mod (n, phil)))
              else proper-phil (state,
                                phil,
                                add1-mod (n, phil)) endif
          elseif sub1-mod (n, phil) < index
          then proper-phil (state,
                            sub1-mod (n, phil),
                            add1-mod (n, sub1-mod (n, phil)))
          else t endif))
```

THEOREM: proper-phils-preserved

```
((1 < n)
 ∧  (statement ∈ phil-prg (n))
 ∧  n (old, new, statement)
 ∧  proper-phils (old, n))
→  proper-phils (new, n)
```

EVENT: Disable but-triple-and-triple-all.

EVENT: Disable but-triple-preserved.

EVENT: Disable proper-triple-preserved.

EVENT: Disable proper-phils.

THEOREM: proper-phils-unless-sufficient

```
(1 < n)
→  unless-sufficient (statement,
                      phil-prg (n),
                      old,
                      new,
                      `(proper-phils state ',n),
                      '(false))
```

THEOREM: all-lefts-unchanged

(all-lefts $(old,\ n)$ $\wedge$ changed $(old,\ new,$ **nil**$))$ $\rightarrow$ all-lefts $(new,\ n)$

THEOREM: all-lefts-unless-sufficient

```
(1 < n)
→  unless-sufficient (statement,
                      phil-prg (n),
                      old,
```

new,
`` `(all-lefts state ',n) ``,
`'(false)`)

EVENT: Disable all-lefts-implies.

EVENT: Disable all-lefts-unchanged.

THEOREM: all-rights-rec-unchanged
(all-rights-rec (*old*, *index*, *n*) $\wedge$ changed (*old*, *new*, **nil**) $\wedge$ ($n \not< index$))
$\rightarrow$ all-rights-rec (*new*, *index*, *n*)

THEOREM: all-rights-unless-sufficient
($1 < n$)
$\rightarrow$ unless-sufficient (*statement*,
phil-prg (n),
old,
new,
`` `(all-rights state ',n) ``,
`'(false)`)

EVENT: Disable all-rights-implies.

EVENT: Disable all-rights-rec-unchanged.

THEOREM: proper-forks-rec-inv
($1 < n$)
$\rightarrow$ unless (`` `(proper-forks-rec state ',n ',n) ``,
`'(false)`,
phil-prg (n))

THEOREM: proper-forks-inv
($1 < n$)
$\rightarrow$ unless (`` `(proper-forks state ',n) ``, `'(false)`, phil-prg (n))

THEOREM: proper-phils-inv
($1 < n$)
$\rightarrow$ unless (`` `(proper-phils state ',n) ``, `'(false)`, phil-prg (n))

THEOREM: all-lefts-inv
($1 < n$) $\rightarrow$ unless (`` `(all-lefts state ',n) ``, `'(false)`, phil-prg (n))

THEOREM: all-rights-inv
($1 < n$)
$\rightarrow$ unless (`` `(all-rights state ',n) ``, `'(false)`, phil-prg (n))

THEOREM: phil-prg-invariant-1

```
((1 < n)
 ∧  initial-condition (‘(and
                          (proper-phils state ’,n)
                          (proper-forks state ’,n)),
                        phil-prg (n)))
 →  (invariant (‘(proper-phils state ’,n), phil-prg (n))
      ∧  invariant (‘(proper-forks state ’,n), phil-prg (n))
      ∧  invariant (‘(and
                         (proper-phils state ’,n)
                         (proper-forks state ’,n)),
                       phil-prg (n)))
```

THEOREM: proper-phil-invariant

```
((index < n)
 ∧  (index ∈ N)
 ∧  (1 < n)
 ∧  initial-condition (‘(and
                          (proper-phils state ’,n)
                          (proper-forks state ’,n)),
                        phil-prg (n)))
 →  invariant (‘(proper-phil state
                   ’,index
                   ’,(add1-mod n index)),
                 phil-prg (n))
```

DEFINITION:

```
proper-fork (state, index, left)
=   (free (state, index)
      ∨  (fork (state, index) = index)
      ∨  (fork (state, index) = left))
```

THEOREM: proper-fork-invariant

```
((index < n)
 ∧  (index ∈ N)
 ∧  (1 < n)
 ∧  initial-condition (‘(and
                          (proper-phils state ’,n)
                          (proper-forks state ’,n)),
                        phil-prg (n)))
 →  invariant (‘(proper-fork state
                   ’,index
                   ’,(sub1-mod n index)),
                 phil-prg (n))
```

THEOREM: hungry-unless-owns-left
$((1 < n) \wedge (index < n) \wedge (index \in \mathbf{N}))$
$\rightarrow$ unless (`` `(hungry state ',index) ``,
`` `(owns-left state ',index) ``,
phil-prg (n))

THEOREM: hungry-unless-owns-right
$((1 < n) \wedge (index < n) \wedge (index \in \mathbf{N}))$
$\rightarrow$ unless (`` `(hungry state ',index) ``,
`` `(owns-right state ',index ',n) ``,
phil-prg (n))

THEOREM: hungry-left-free-e-ensures-owns-left
$((1 < n) \wedge (index < n) \wedge (index \in \mathbf{N}))$
$\rightarrow$ e-ensures (`` `(hungry state ',index) ``,

```
`(owns-left state ',index),
`(and
  (hungry state ',index)
  (free state ',index)),
```

phil-prg (n))

THEOREM: hungry-right-free-e-ensures-owns-right
$((1 < n) \wedge (index < n) \wedge (index \in \mathbf{N}))$
$\rightarrow$ e-ensures (`` `(hungry state ',index) ``,

```
`(owns-right state ',index ',n),
`(and
  (hungry state ',index)
  (free state ',(add1-mod n index))),
```

phil-prg (n))

THEOREM: hungry-unless-eating
$((1 < n) \wedge (index < n) \wedge (index \in \mathbf{N}))$
$\rightarrow$ unless (`` `(hungry state ',index) ``,
`` `(eating state ',index) ``,
phil-prg (n))

THEOREM: owns-left-unless-sufficient
$((1 < n) \wedge (index < n) \wedge (index \in \mathbf{N}))$
$\rightarrow$ unless-sufficient (*statement*,
phil-prg (n),
old,
new,

```
`(and
  (proper-phils state ',n)
  (owns-left state ',index)),
`(eating state ',index))
```

THEOREM: owns-right-unless-sufficient
$((1 < n) \wedge (index < n) \wedge (index \in \mathbf{N}))$
$\rightarrow$ unless-sufficient (*statement*,
phil-prg (n),
old,
new,
```
'(and
  (proper-phils state ',n)
  (owns-right state ',index ',n)),
'(eating state ',index))
```

THEOREM: owns-right-unless-eating
$((1 < n) \wedge (index < n) \wedge (index \in \mathbf{N}))$
$\rightarrow$ unless (
```
'(and
   (proper-phils state ',n)
   (owns-right state ',index ',n)),
'(eating state ',index),
```
phil-prg (n))

THEOREM: owns-left-unless-eating
$((1 < n) \wedge (index < n) \wedge (index \in \mathbf{N}))$
$\rightarrow$ unless (
```
'(and
   (proper-phils state ',n)
   (owns-left state ',index)),
'(eating state ',index),
```
phil-prg (n))

THEOREM: owns-both-unless-eating
$((1 < n) \wedge (index < n) \wedge (index \in \mathbf{N}))$
$\rightarrow$ unless (
```
'(and
   (owns-left state ',index)
   (and
    (owns-right state ',index ',n)
    (proper-phils state ',n))),
'(eating state ',index),
```
phil-prg (n))

THEOREM: owns-both-e-ensures-eating
$((1 < n) \wedge (index < n) \wedge (index \in \mathbf{N}))$
$\rightarrow$ e-ensures (
```
'(and
   (owns-left state ',index)
   (and
    (owns-right state ',index ',n)
    (proper-phils state ',n))),
'(eating state ',index),
```

```
                  `(true),
                  phil-prg (n))
```

THEOREM: owns-both-leads-to-eating

```
 ((1 < n)
  ∧   (index < n)
  ∧   (index ∈ N)
  ∧   initial-condition (`(and
                             (proper-phils state ',n)
                             (proper-forks state ',n)),
                          phil-prg (n)))
 →   leads-to (`(and
                  (owns-left state ',index)
                  (owns-right state ',index ',n)),
               `(eating state ',index),
               phil-prg (n))
```

THEOREM: eating-unless-free

```
 ((1 < n) ∧ (index < n) ∧ (index ∈ N))
 →   unless (`(eating state ',index),
             `(and
               (free state ',index)
               (free state ',(add1-mod n index))),
             phil-prg (n))
```

THEOREM: eating-e-ensures-free

```
 ((1 < n) ∧ (index < n) ∧ (index ∈ N))
 →   e-ensures (`(eating state ',index),
                `(and
                  (free state ',index)
                  (free state ',(add1-mod n index))),
                '(true),
                phil-prg (n))
```

THEOREM: eating-leads-to-free

```
 ((1 < n)
  ∧   (index < n)
  ∧   (index ∈ N)
  ∧   initial-condition (`(and
                             (proper-phils state ',n)
                             (proper-forks state ',n)),
                          phil-prg (n)))
 →   leads-to (`(eating state ',index),
               `(and
                  (free state ',index)
```

```
            (free state ',(add1-mod n index))),
          phil-prg (n))
```

THEOREM: leads-to-expanded-right-implies

```
((1 < n)
 ∧  (index < n)
 ∧  (index ∈ N)
 ∧  initial-condition (‘(and
                         (proper-phils state ',n)
                         (proper-forks state ',n)),
                       phil-prg (n))
 ∧  leads-to (‘(true),
              ‘(or
                (owns-right state ',(sub1-mod n index) ',n)
                (or
                 (free state ',index)
                 (or
                  (eating state ',index)
                  (thinking state ',index)))),
              phil-prg (n)))
→   leads-to (‘(true),
              ‘(or
                (free state ',index)
                (owns-right state ',(sub1-mod n index) ',n)),
              phil-prg (n))
```

THEOREM: leads-to-expanded-left-implies

```
((1 < n)
 ∧  (index < n)
 ∧  (index ∈ N)
 ∧  initial-condition (‘(and
                         (proper-phils state ',n)
                         (proper-forks state ',n)),
                       phil-prg (n))
 ∧  leads-to (‘(true),
              ‘(or
                (owns-left state ',(add1-mod n index))
                (or
                 (free state ',(add1-mod n index))
                 (or
                  (eating state ',index)
                  (thinking state ',index)))),
              phil-prg (n)))
→   leads-to (‘(true),
```

```
              `(or
                (free state ',(add1-mod n index))
                (owns-left state ',(add1-mod n index))),
              phil-prg (n))
```

THEOREM: true-leads-to-left-free-implies

```
((1 < n)
 ∧  (index < n)
 ∧  (index ∈ N)
 ∧  strongly-fair (phil-prg (n))
 ∧  initial-condition (`(and
                          (proper-phils state ',n)
                          (proper-forks state ',n)),
                        phil-prg (n))
 ∧  leads-to (`(true),
              `(or
                (free state ',index)
                (owns-left state ',index)),
              phil-prg (n)))
→   leads-to (`(hungry state ',index),
              `(owns-left state ',index),
              phil-prg (n))
```

THEOREM: true-leads-to-right-free-implies

```
((1 < n)
 ∧  (index < n)
 ∧  (index ∈ N)
 ∧  strongly-fair (phil-prg (n))
 ∧  initial-condition (`(and
                          (proper-phils state ',n)
                          (proper-forks state ',n)),
                        phil-prg (n))
 ∧  leads-to (`(true),
              `(or
                (free state ',(add1-mod n index))
                (owns-right state ',index ',n)),
              phil-prg (n)))
→   leads-to (`(hungry state ',index),
              `(owns-right state ',index ',n),
              phil-prg (n))
```

THEOREM: hungry-leads-to-owns-right-implies

```
((1 < n)
 ∧  (index < n)
 ∧  (index ∈ N)
```

$\wedge$ strongly-fair (phil-prg (n))
$\wedge$ initial-condition (`‘(and`
```
                 (proper-phils state ',n)
                 (proper-forks state ',n)),
```
phil-prg (n))
$\wedge$ leads-to (`‘(hungry state ',index)`,
`‘(owns-right state ',index ',n)`,
phil-prg (n)))
$\rightarrow$ leads-to (`‘(and`
```
      (hungry state ',index)
      (owns-left state ',index)),
    ‘(eating state ',index),
```
phil-prg (n))

THEOREM: hungry-leads-to-owns-left-implies
$((1 < n)$
$\wedge$ $(index < n)$
$\wedge$ $(index \in \mathbf{N})$
$\wedge$ strongly-fair (phil-prg (n))
$\wedge$ initial-condition (`‘(and`
```
                 (proper-phils state ',n)
                 (proper-forks state ',n)),
```
phil-prg (n))
$\wedge$ leads-to (`‘(hungry state ',index)`,
`‘(owns-left state ',index)`,
phil-prg (n)))
$\rightarrow$ leads-to (`‘(and`
```
      (hungry state ',index)
      (owns-right state ',index ',n)),
    ‘(eating state ',index),
```
phil-prg (n))

DEFINITION:
left-chain (i, j, n, $state$)
= **if** $(i \in \mathbf{N}) \wedge (j \in \mathbf{N}) \wedge (i < n) \wedge (j < n)$
then if $i = j$ **then** hungry ($state$, i) $\wedge$ owns-left ($state$, i)
else hungry ($state$, i)
$\wedge$ owns-left ($state$, i)
$\wedge$ left-chain (sub1-mod (n, i), j, n, $state$) **endif**
else f endif

DEFINITION:
right-chain (i, j, n, $state$)
= **if** $(i \in \mathbf{N}) \wedge (j \in \mathbf{N}) \wedge (i < n) \wedge (j < n)$
then if $i = j$ **then** hungry ($state$, i) $\wedge$ owns-right ($state$, i, n)

$\quad\quad$ **else** hungry $(state,\ i)$

$\quad\quad\quad \land \quad$ owns-right $(state,\ i,\ n)$

$\quad\quad\quad \land \quad$ right-chain (add1-mod $(n,\ i),\ j,\ n,\ state)$ **endif**

$\quad$ **else f endif**

THEOREM: break-left-chain

$((i < n) \land (j < n) \land (i \in \mathbf{N}) \land (j \in \mathbf{N}) \land (i < j))$

$\rightarrow \quad$ (left-chain $(i,\ j,\ n,\ state)$

$\quad\quad = \quad$ (left-chain $(i,\ \mathtt{0},\ n,\ state) \land$ left-chain $(n - 1,\ j,\ n,\ state)))$

DEFINITION:

lcc $(i,\ j,\ k)$

$= \quad$ **if** $(i \in \mathbf{N})$

$\quad\quad \land \quad (j \in \mathbf{N})$

$\quad\quad \land \quad (k \in \mathbf{N})$

$\quad\quad \land \quad (i < k)$

$\quad\quad \land \quad (j < k)$

$\quad\quad \land \quad (i \neq j)$

$\quad\quad \land \quad (k \not\simeq \mathtt{0})$

$\quad$ **then if** $(1 + i) = k$ **then** lcc $(i - 1,\ j,\ k - 1)$

$\quad\quad$ **elseif** $(1 + j) = k$ **then** lcc $(i,\ j - 1,\ k - 1)$

$\quad\quad$ **else** lcc $(i,\ j,\ k - 1)$ **endif**

$\quad$ **else t endif**

THEOREM: left-chain-combine

$((i \in \mathbf{N}) \land (j \in \mathbf{N}) \land (k \in \mathbf{N}) \land (k < n) \land (i < k) \land (j < k))$

$\rightarrow \quad$ ((left-chain $(i,\ \mathtt{0},\ n,\ state) \land$ left-chain $(k,\ 1 + i,\ n,\ state))$

$\quad\quad = \quad$ (left-chain $(j,\ \mathtt{0},\ n,\ state) \land$ left-chain $(k,\ 1 + j,\ n,\ state)))$

THEOREM: lessp-1-1

$(n < \mathtt{1}) = (n \simeq \mathtt{0})$

EVENT: Disable lessp-1-1.

THEOREM: left-chain-canonicalize

$((i \in \mathbf{N}) \land (i < n))$

$\rightarrow \quad$ (left-chain $(i,$ add1-mod $(n,\ i),\ n,\ state) =$ left-chain $(n - 1,\ \mathtt{0},\ n,\ state))$

THEOREM: left-chain-canonicalize-all-lefts

$((index \not\simeq \mathtt{0}) \land (n \not< index))$

$\rightarrow \quad$ (left-chain $(index - 1,\ \mathtt{0},\ n,\ state) =$ all-lefts $(state,\ index))$

THEOREM: left-chain-complete

$((i \in \mathbf{N}) \land (i < n))$

$\rightarrow \quad$ (left-chain $(i,$ add1-mod $(n,\ i),\ n,\ state) =$ all-lefts $(state,\ n))$

EVENT: Disable left-chain-canonicalize-all-lefts.

EVENT: Disable left-chain-canonicalize.

EVENT: Disable left-chain-combine.

EVENT: Disable break-left-chain.

THEOREM: break-right-chain
$((i < n) \land (j < n) \land (i \in \mathbf{N}) \land (j \in \mathbf{N}) \land (j < i))$
$\rightarrow$ (right-chain $(i, j, n, \mathit{state})$
$\quad = \quad$ (right-chain $(i, n-1, n, \mathit{state}) \land$ right-chain $(\mathtt{0}, j, n, \mathit{state})))$

DEFINITION:
rcc (i, j, k, n)
$=$ **if** $(i < n)$
$\quad\land\quad (j < n)$
$\quad\land\quad (k \in \mathbf{N})$
$\quad\land\quad (k < i)$
$\quad\land\quad (k < j)$
$\quad\land\quad (i \neq j)$
$\quad\land\quad (k \neq n)$
then if $k = (i-1)$ **then** rcc $(1+i, j, 1+k, n)$
elseif $k = (j-1)$ **then** rcc $(i, 1+j, 1+k, n)$
else rcc $(i, j, 1+k, n)$ **endif**
else t endif

THEOREM: right-chain-combine
$((k \in \mathbf{N}) \land (i < n) \land (j < n) \land (k < i) \land (k < j))$
$\rightarrow$ ((right-chain $(i, n-1, n, \mathit{state}) \land$ right-chain $(k, i-1, n, \mathit{state}))$
$\quad = \quad$ (right-chain $(j, n-1, n, \mathit{state})$
$\quad\quad \land \quad$ right-chain $(k, j-1, n, \mathit{state})))$

THEOREM: right-chain-canonicalize
$((i \in \mathbf{N}) \land (i < n))$
$\rightarrow$ (right-chain $(i,$ sub1-mod $(n, i), n, \mathit{state})$
$\quad = \quad$ right-chain $(\mathtt{0}, n-1, n, \mathit{state}))$

THEOREM: right-chain-extend-right
$((i \in \mathbf{N}) \land (j \in \mathbf{N}) \land (i < n) \land (j < n) \land (i \neq j))$
$\rightarrow$ (right-chain $(i, j, n, \mathit{state})$
$\quad = \quad$ (right-chain $(i,$ sub1-mod $(n, j), n, \mathit{state})$
$\quad\quad \land \quad$ hungry (state, j)
$\quad\quad \land \quad$ owns-right $(\mathit{state}, j, n)))$

THEOREM: right-chain-canonicalize-all-rights
$((index \not\simeq \mathtt{0}) \land (n \not< index))$
$\rightarrow$ (right-chain ($\mathtt{0}$, $index - 1$, n, $state$) = all-rights-rec ($state$, $index$, n))

THEOREM: right-chain-complete
$((i \in \mathbf{N}) \land (i < n))$
$\rightarrow$ (right-chain (i, sub1-mod (n, i), n, $state$) = all-rights ($state$, n))

EVENT: Disable right-chain-canonicalize-all-rights.

EVENT: Disable right-chain-extend-right.

EVENT: Disable right-chain-canonicalize.

EVENT: Disable right-chain-combine.

EVENT: Disable break-right-chain.

THEOREM: not-eventually-left-implies
$((\mathtt{1} < n)$
$\land$ $(index < n)$
$\land$ $(index \in \mathbf{N})$
$\land$ initial-condition (`(and
```
                  (proper-phils state ',n)
                  (proper-forks state ',n)),
```
phil-prg (n))
$\land$ ($\neg$ eventually-stable (`(and
```
                     (hungry state ',index)
                     (owns-left state ',index)),
```
phil-prg (n))))
$\rightarrow$ leads-to (`(true),
```
          `(or
            (owns-right state ',(sub1-mod n index) ',n)
            (or
             (free state ',index)
             (or
              (eating state ',index)
              (thinking state ',index)))),
```
phil-prg (n))

THEOREM: not-eventually-right-implies
$((\mathtt{1} < n)$

```
 ∧   (index < n)
 ∧   (index ∈ N)
 ∧   initial-condition (‘(and
                          (proper-phils state ’,n)
                          (proper-forks state ’,n)),
                        phil-prg (n))
 ∧   (¬ eventually-stable (‘(and
                              (hungry state ’,index)
                              (owns-right state ’,index ’,n)),
                            phil-prg (n))))
→   leads-to (‘(true),
              ‘(or
                 (owns-left state ’,(add1-mod n index))
                 (or
                  (free state ’,(add1-mod n index))
                  (or
                   (eating state ’,index)
                   (thinking state ’,index)))),
              phil-prg (n))
```

THEOREM: right-chain-induction

```
 (strongly-fair (phil-prg (n))
 ∧   (1 < n)
 ∧   (i < n)
 ∧   (j < n)
 ∧   (i ∈ N)
 ∧   (j ∈ N)
 ∧   initial-condition (‘(and
                          (proper-phils state ’,n)
                          (proper-forks state ’,n)),
                        phil-prg (n))
 ∧   eventually-stable (‘(and
                           (hungry state ’,j)
                           (owns-right state ’,j ’,n)),
                         phil-prg (n)))
→   eventually-stable (‘(right-chain ’,i ’,j ’,n state),
                        phil-prg (n))
```

THEOREM: eventually-stable-right-implies-all-rights

```
 ((1 < n)
 ∧   (j < n)
 ∧   (j ∈ N)
 ∧   strongly-fair (phil-prg (n))
 ∧   initial-condition (‘(and
```

(proper-phils state ',n)
(proper-forks state ',n)),
phil-prg (n))
$\wedge$ eventually-stable (`(and
(hungry state ',j)
(owns-right state ',j ',n)),
phil-prg (n)))
$\rightarrow$ eventually-stable (`(all-rights state ',n), phil-prg (n))

THEOREM: never-all-rights
(deadlock-free (phil-prg (n)) $\wedge$ $(1 < n)$)
$\rightarrow$ invariant (`(not (all-rights state ',n)), phil-prg (n))

THEOREM: not-eventually-owns-right
($(1 < n)$
$\wedge$ $(j < n)$
$\wedge$ $(j \in \mathbf{N})$
$\wedge$ strongly-fair (phil-prg (n))
$\wedge$ deadlock-free (phil-prg (n))
$\wedge$ initial-condition (`(and
(proper-phils state ',n)
(proper-forks state ',n)),
phil-prg (n)))
$\rightarrow$ ($\neg$ eventually-stable (`(and
(hungry state ',j)
(owns-right state ',j ',n)),
phil-prg (n))))

THEOREM: hungry-leads-to-owns-left
($(1 < n)$
$\wedge$ $(index \in \mathbf{N})$
$\wedge$ $(index < n)$
$\wedge$ initial-condition (`(and
(proper-phils state ',n)
(proper-forks state ',n)),
phil-prg (n))
$\wedge$ strongly-fair (phil-prg (n))
$\wedge$ deadlock-free (phil-prg (n)))
$\rightarrow$ leads-to (`(hungry state ',index),
`(owns-left state ',index),
phil-prg (n))

THEOREM: left-chain-induction
(strongly-fair (phil-prg (n))
$\wedge$ $(1 < n)$

$\wedge$ $(i < n)$
$\wedge$ $(j < n)$
$\wedge$ $(i \in \mathbf{N})$
$\wedge$ $(j \in \mathbf{N})$
$\wedge$ initial-condition (`` `(and ``
`(proper-phils state ',n)`
`(proper-forks state ',n))`,
phil-prg (n))
$\wedge$ eventually-stable (`` `(and ``
`(hungry state ',j)`
`(owns-left state ',j))`,
phil-prg (n)))
$\rightarrow$ eventually-stable (`` `(left-chain ',i ',j ',n state) ``,
phil-prg (n))

THEOREM: eventually-stable-left-implies-all-lefts
$((1 < n)$
$\wedge$ $(j < n)$
$\wedge$ $(j \in \mathbf{N})$
$\wedge$ strongly-fair (phil-prg (n))
$\wedge$ initial-condition (`` `(and ``
`(proper-phils state ',n)`
`(proper-forks state ',n))`,
phil-prg (n))
$\wedge$ eventually-stable (`` `(and ``
`(hungry state ',j)`
`(owns-left state ',j))`,
phil-prg (n)))
$\rightarrow$ eventually-stable (`` `(all-lefts state ',n) ``, phil-prg (n))

THEOREM: never-all-lefts
(deadlock-free (phil-prg (n)) $\wedge$ $(1 < n)$)
$\rightarrow$ invariant (`` `(not (all-lefts state ',n)) ``, phil-prg (n))

THEOREM: not-eventually-owns-left
$((1 < n)$
$\wedge$ $(j < n)$
$\wedge$ $(j \in \mathbf{N})$
$\wedge$ strongly-fair (phil-prg (n))
$\wedge$ deadlock-free (phil-prg (n))
$\wedge$ initial-condition (`` `(and ``
`(proper-phils state ',n)`
`(proper-forks state ',n))`,
phil-prg (n)))
$\rightarrow$ ($\neg$ eventually-stable (`` `(and ``

```
                    (hungry state ',j)
                    (owns-left state ',j)),
                  phil-prg (n)))
```

THEOREM: hungry-leads-to-owns-right

```
((1 < n)
 ∧  (index ∈ N)
 ∧  (index < n)
 ∧  initial-condition ('(and
                          (proper-phils state ',n)
                          (proper-forks state ',n)),
                        phil-prg (n))
 ∧  strongly-fair (phil-prg (n))
 ∧  deadlock-free (phil-prg (n)))
→   leads-to ('(hungry state ',index),
              '(owns-right state ',index ',n),
              phil-prg (n))
```

THEOREM: correctness

```
((1 < n)
 ∧  (index ∈ N)
 ∧  (index < n)
 ∧  initial-condition ('(and
                          (proper-phils state ',n)
                          (proper-forks state ',n)),
                        phil-prg (n))
 ∧  strongly-fair (phil-prg (n))
 ∧  deadlock-free (phil-prg (n)))
→   leads-to ('(hungry state ',index),
              '(eating state ',index),
              phil-prg (n))
```

Index

add1-mod, 1, 2, 4–8, 17
all-lefts, 5, 8, 17
all-lefts-implies, 5
all-lefts-inv, 9
all-lefts-unchanged, 8
all-lefts-unless-sufficient, 8
all-rights, 5, 19
all-rights-implies, 5
all-rights-inv, 9
all-rights-rec, 5, 9, 19
all-rights-rec-implies, 5
all-rights-rec-unchanged, 9
all-rights-unless-sufficient, 9

break-left-chain, 17
break-right-chain, 18
but-triple, 6, 7
but-triple-and-triple-all, 7
but-triple-preserved, 7

changed, 1, 2, 8, 9
correctness, 23

deadlock-free, 21–23

e-ensures, 11, 13
eating, 1–4
eating-e-ensures-free, 13
eating-leads-to-free, 13
eating-to, 2
eating-unless-free, 13
eventually-stable, 19–23
eventually-stable-left-implies-all-lefts, 22
eventually-stable-right-implies-all-rights, 20

fork, 1, 3, 4, 10
free, 1–3, 10

hungry, 1, 2, 4, 5, 16–18
hungry-both, 2
hungry-leads-to-owns-left, 21
hungry-leads-to-owns-left-implies, 16
hungry-leads-to-owns-right, 23
hungry-leads-to-owns-right-implies, 15
hungry-left, 1
hungry-left-free-e-ensures-owns-left, 11
hungry-right, 1
hungry-right-free-e-ensures-owns-right, 11
hungry-unless-eating, 11
hungry-unless-owns-left, 11
hungry-unless-owns-right, 11

initial-condition, 10, 13–16, 19–23
invariant, 10, 21, 22

lcc, 17
leads-to, 13–16, 19–21, 23
leads-to-expanded-left-implies, 14
leads-to-expanded-right-implies, 14
left-chain, 16, 17
left-chain-canonicalize, 17
left-chain-canonicalize-all-lefts, 17
left-chain-combine, 17
left-chain-complete, 17
left-chain-induction, 21
lessp-1, 6
lessp-1-1, 17
lessp-2, 6
lessp-2-2, 6
listp-phil-prg, 5

member-phil-prg, 3
member-ring, 3

n, 6–8
never-all-lefts, 22
never-all-rights, 21

not-eventually-left-implies, 19
not-eventually-owns-left, 22
not-eventually-owns-right, 21
not-eventually-right-implies, 19

owns-both-e-ensures-eating, 12
owns-both-leads-to-eating, 13
owns-both-unless-eating, 12
owns-left, 1, 2, 4, 5, 16
owns-left-unless-eating, 12
owns-left-unless-sufficient, 11
owns-right, 1, 2, 4, 5, 16–18
owns-right-unless-eating, 12
owns-right-unless-sufficient, 12

pfusi, 5
phil, 2, 3, 6, 7
phil-prg, 2, 3, 5, 6, 8–16, 19–23
phil-prg-invariant-1, 10
proper-fork, 10
proper-fork-invariant, 10
proper-forks, 3
proper-forks-implies, 3
proper-forks-inv, 9
proper-forks-rec, 3
proper-forks-rec-implies, 3
proper-forks-rec-inv, 9
proper-forks-rec-unless-sufficie
nt, 6
proper-phil, 3, 4, 6–8
proper-phil-invariant, 10
proper-phils, 4, 8
proper-phils-implies-proper-phi
l, 4
proper-phils-inv, 9
proper-phils-preserved, 8
proper-phils-rec, 4, 7
proper-phils-rec-implies, 4
proper-phils-rec-implies-proper
-phil, 4
proper-phils-unless-sufficient, 8
proper-triple, 6, 7
proper-triple-preserved, 6

rcc, 18
right-chain, 16–19
right-chain-canonicalize, 18
right-chain-canonicalize-all-ri
ghts, 19
right-chain-combine, 18
right-chain-complete, 19
right-chain-extend-right, 18
right-chain-induction, 20
ring, 2, 3

status, 1
strongly-fair, 15, 16, 20–23
sub1-mod, 3, 6–8, 16, 18, 19

thinking, 1–4
thinking-to, 1
true-leads-to-left-free-implies, 15
true-leads-to-right-free-implie
s, 15

unless, 9, 11–13
unless-sufficient, 6, 8, 9, 11, 12